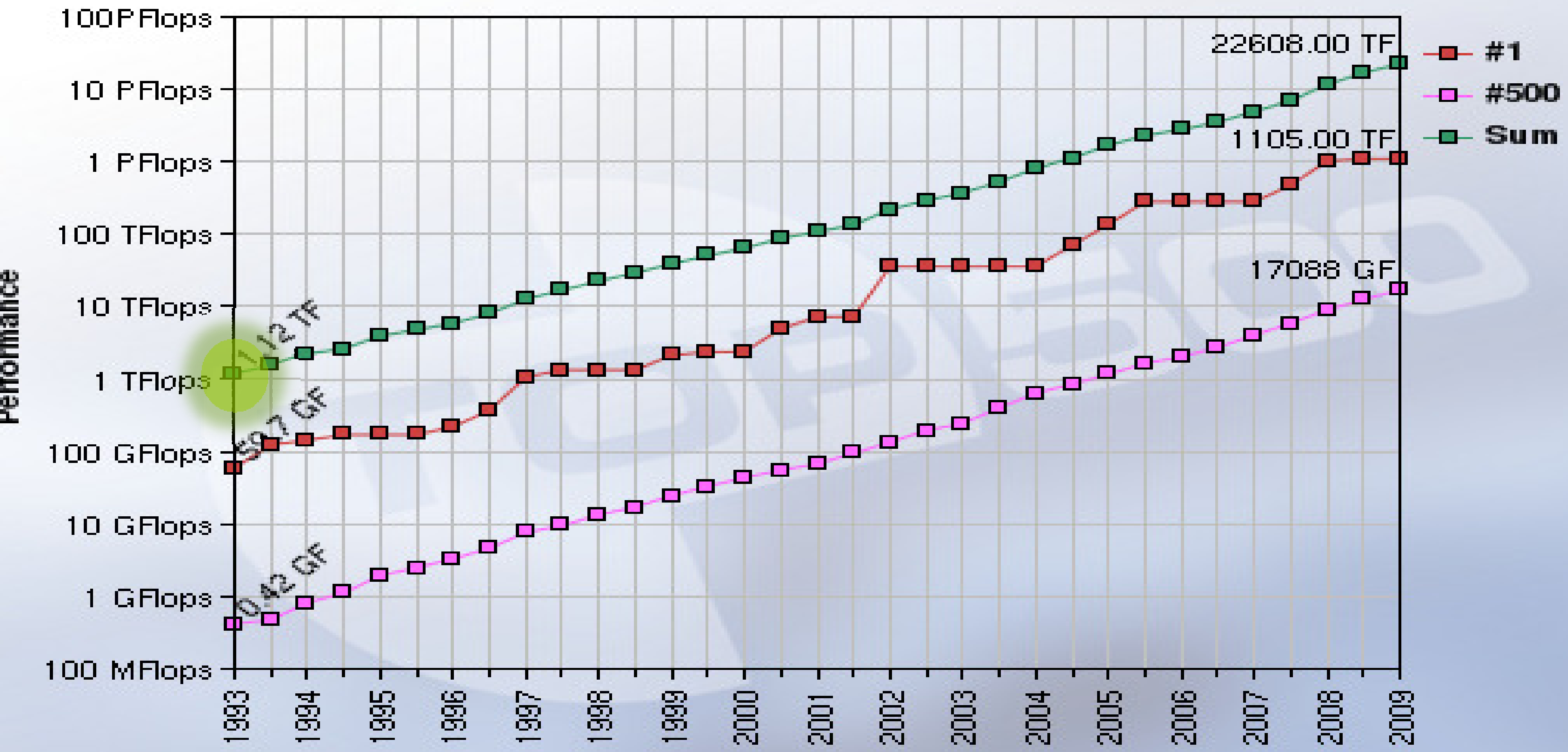
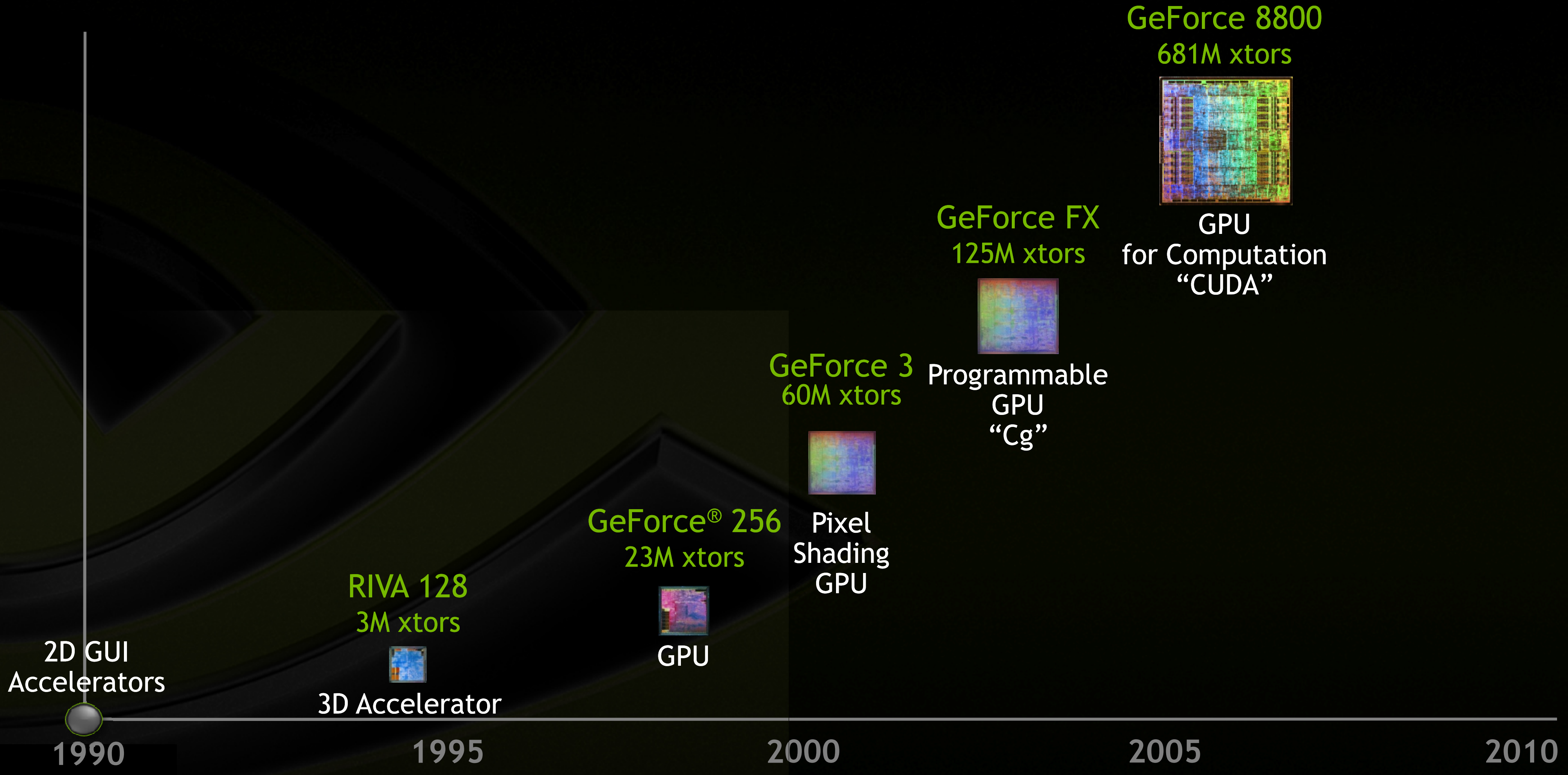
The Power of Heterogeneous Parallel Computing: Computational Graphics

David B. Kirk, NVIDIA Fellow

Performance Development

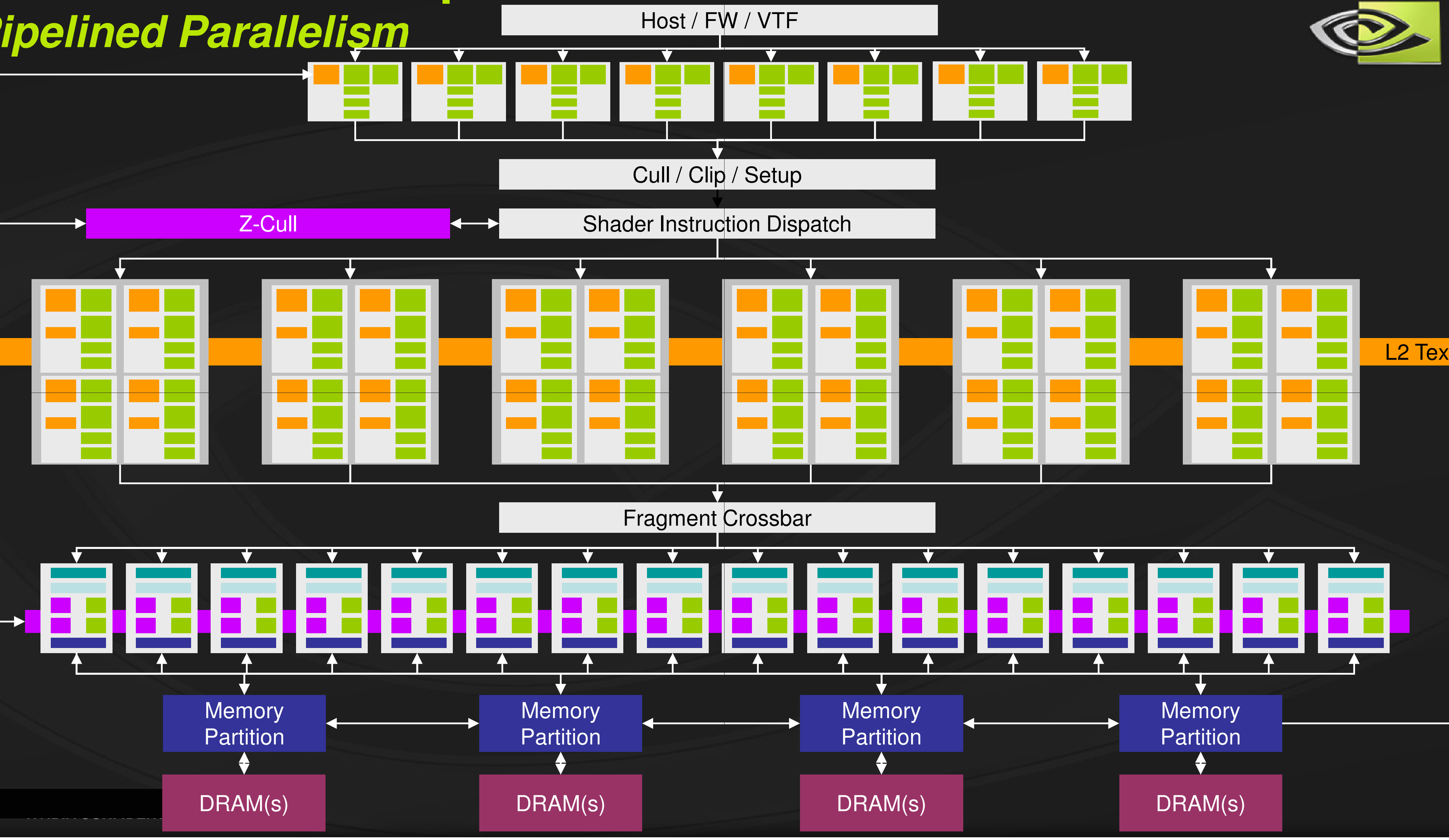


NVIDIA Technology Evolution



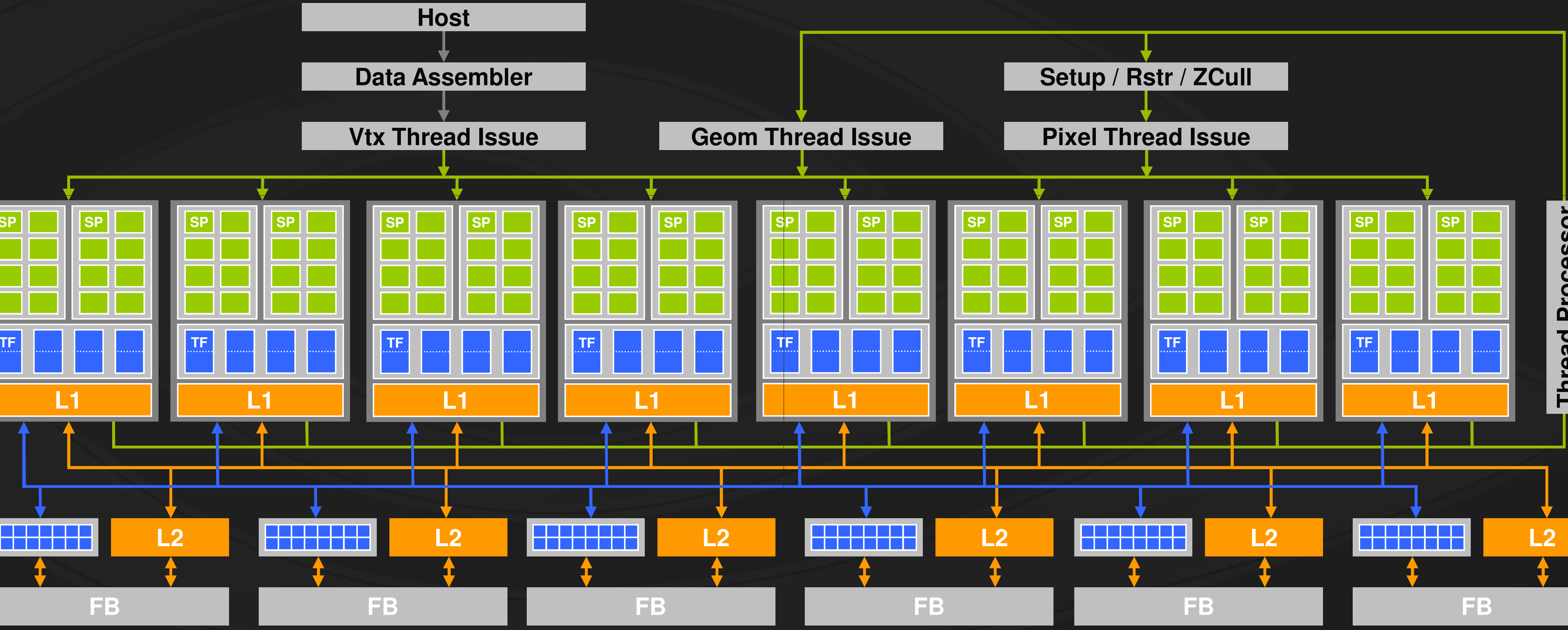
eForce 7800 - Graphics

ipelined Parallelism



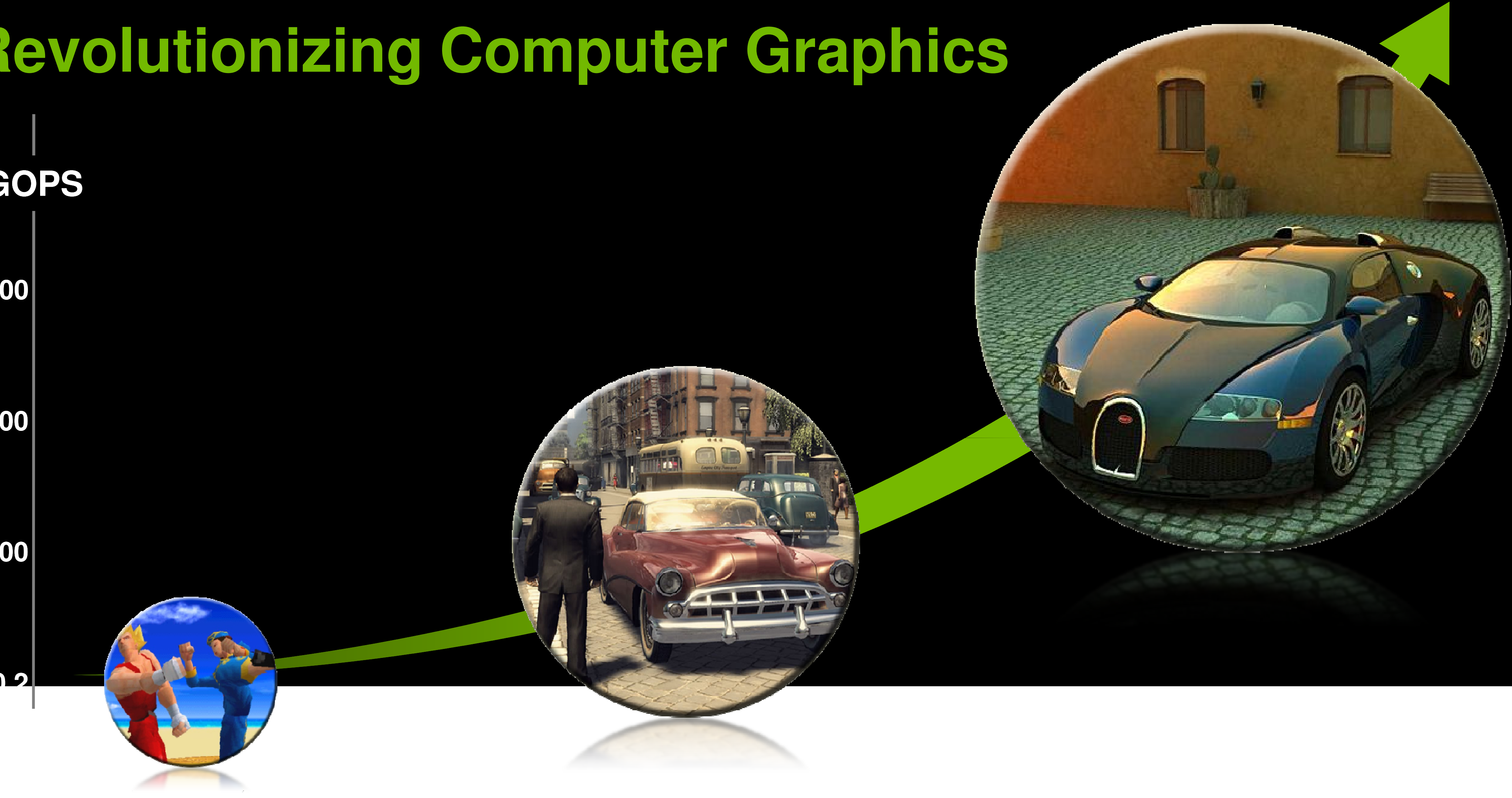
GF8800 replaces the pipeline model

- The future of GPUs is programmable processing
- So – build the architecture around the processor(s)

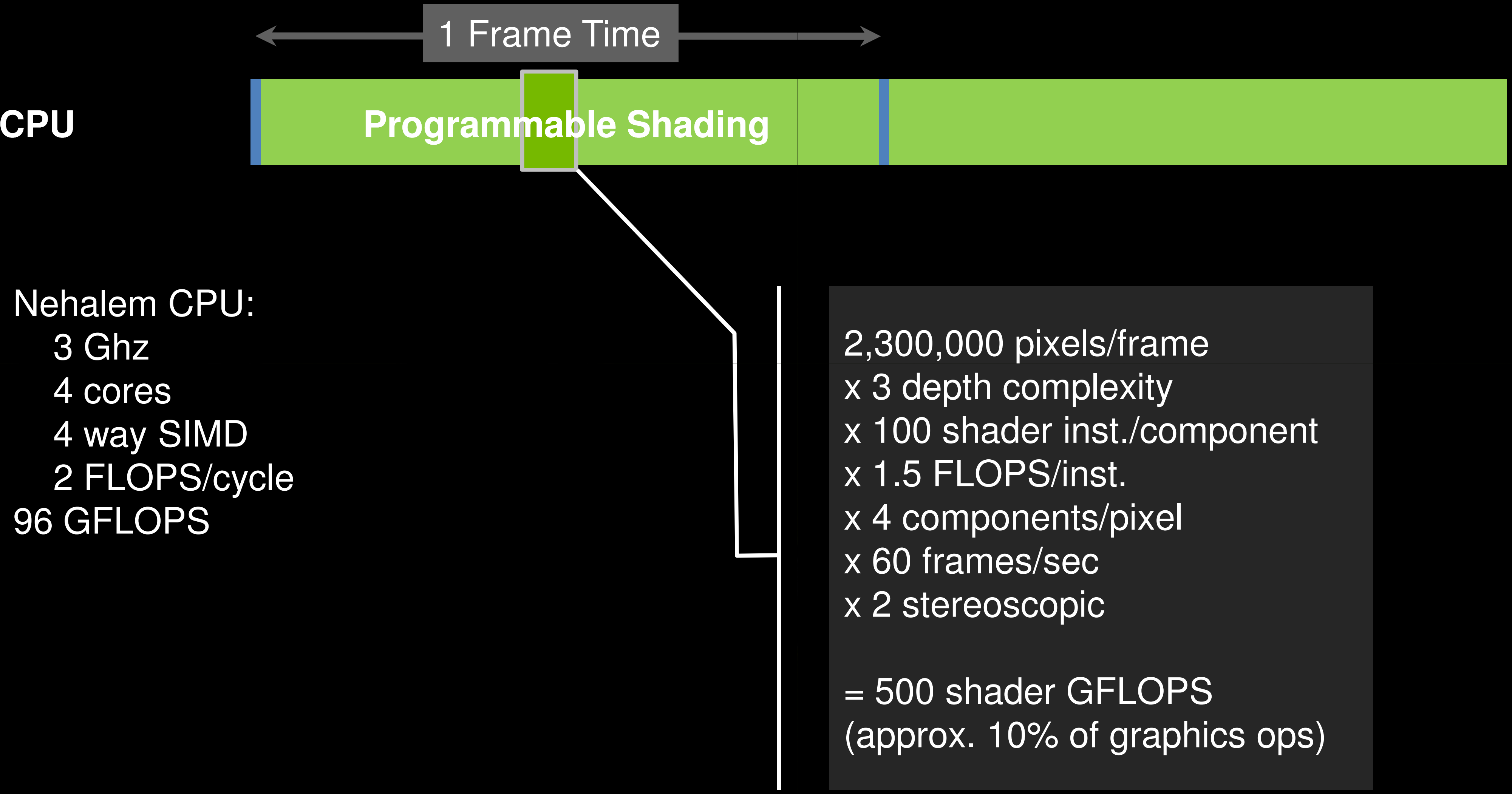


[redacted]

Revolutionizing Computer Graphics



Computer Graphics – A Study in Parallelism



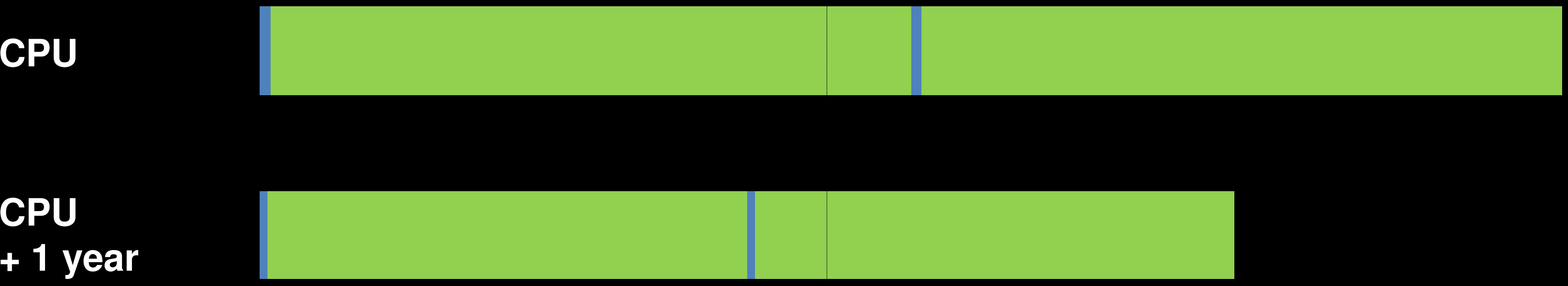
Nehalem CPU:
- 3 Ghz
- 4 cores
- 4 way SIMD
- 2 FLOPS/cycle

96 GFLOPS

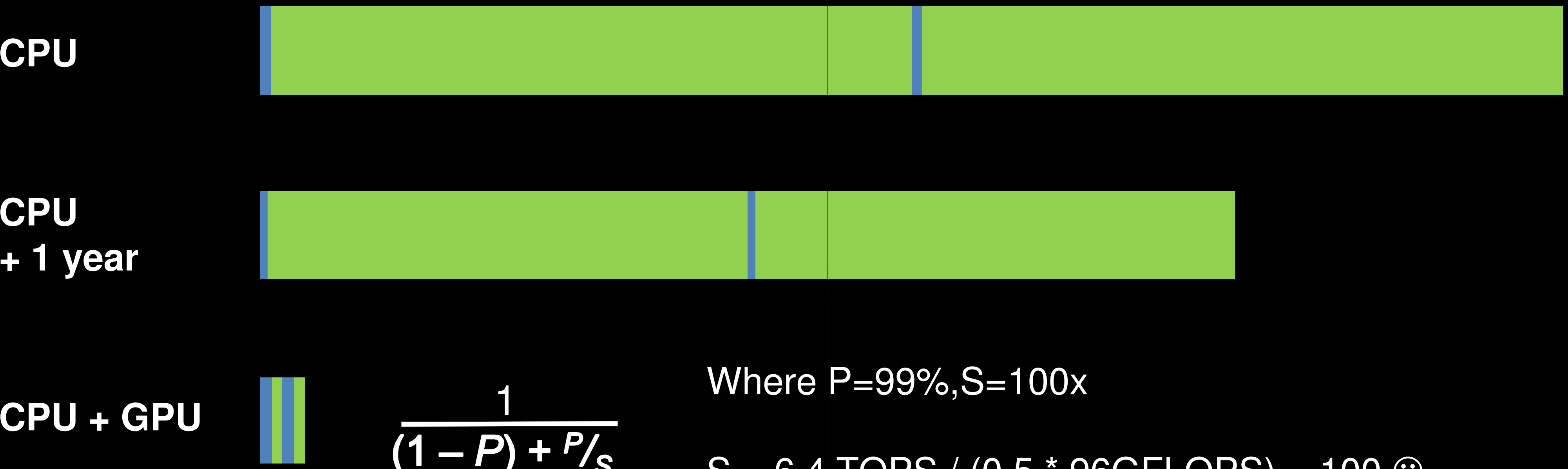
2,300,000 pixels/frame
x 3 depth complexity
x 100 shader inst./component
x 1.5 FLOPS/inst.
x 4 components/pixel
x 60 frames/sec
x 2 stereoscopic

= 500 shader GFLOPS
(approx. 10% of graphics ops)

Computer Graphics – A Study in Parallelism



Computer Graphics – A Study in Parallelism

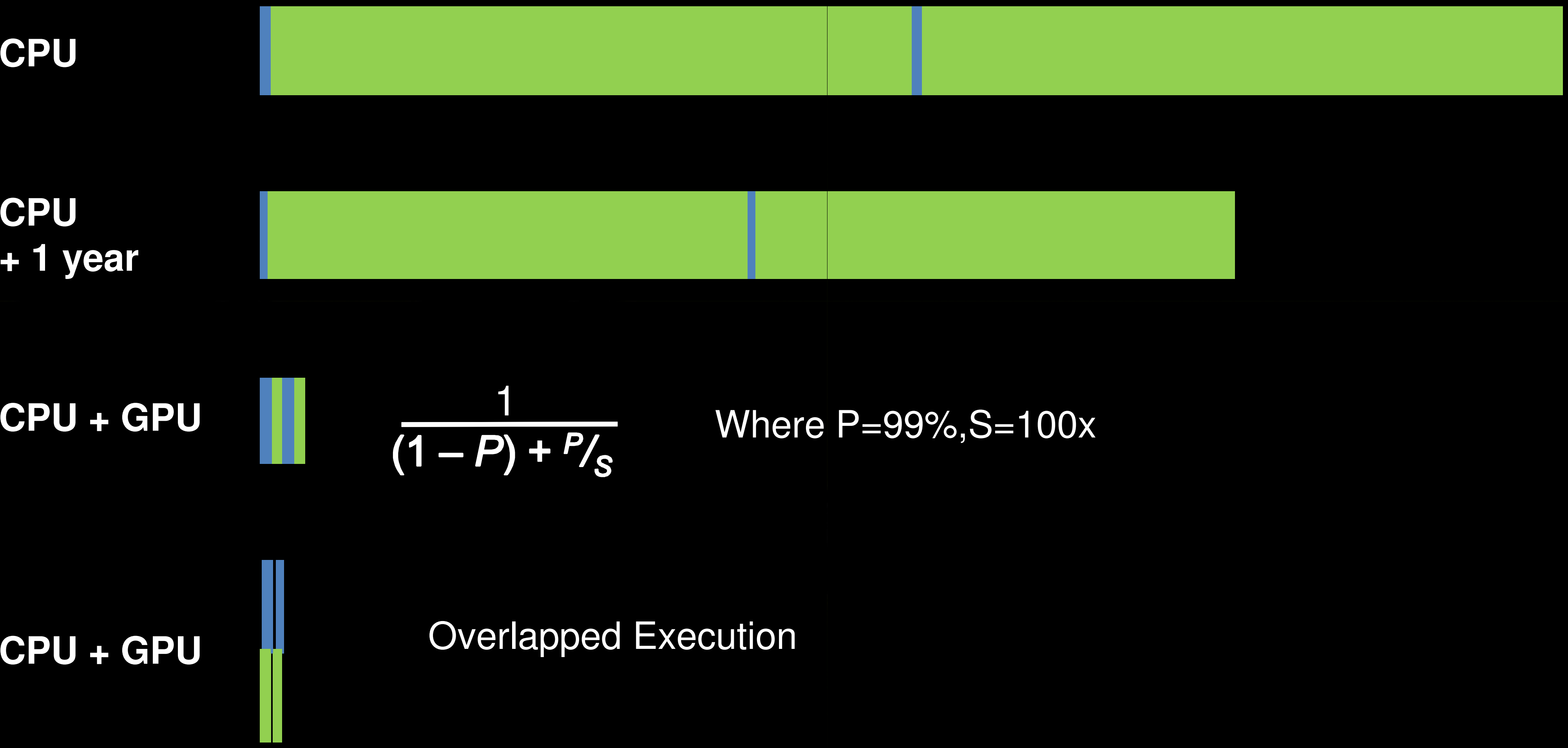


$$\frac{1}{(1 - P) + {}^{P}/_{S}}$$

Where P=99%,S=100x

S = 6.4 TOPS / (0.5 * 96GFLOPS) = 100 ☺

Computer Graphics – A Study in Parallelism



$$\frac{1}{(1-P)+{}^{P}/_{S}}$$

Where P=99%,S=100x

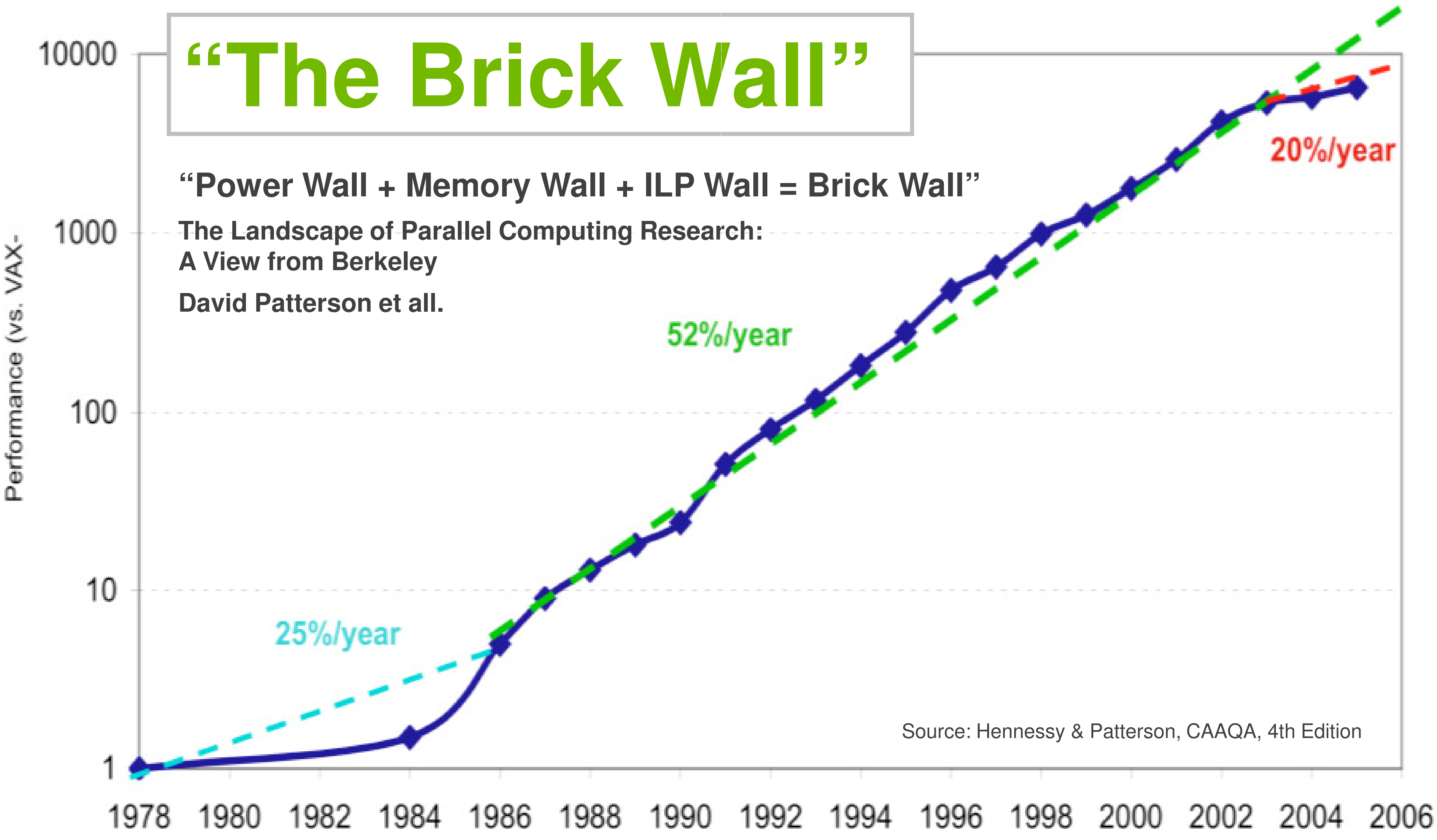
Overlapped Execution

“The Brick Wall”

“Power Wall + Memory Wall + ILP Wall = Brick Wall”

The Landscape of Parallel Computing Research:
A View from Berkeley

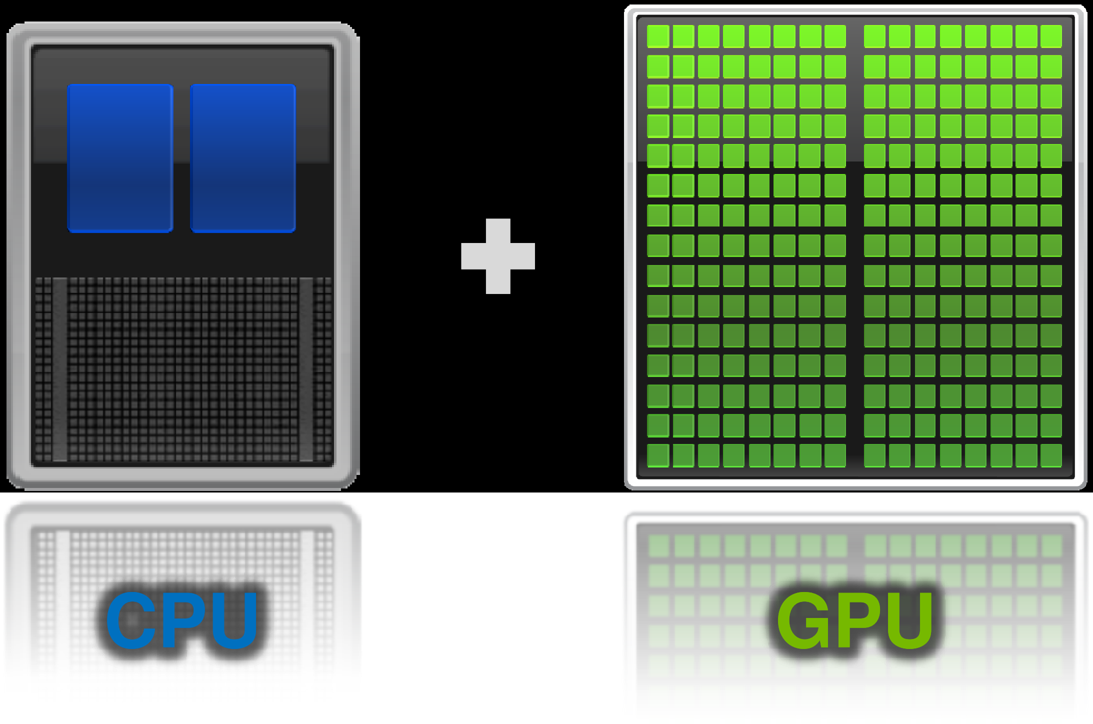
David Patterson et all.



Source: Hennessy & Patterson, CAAQA, 4th Edition

Co-Processing

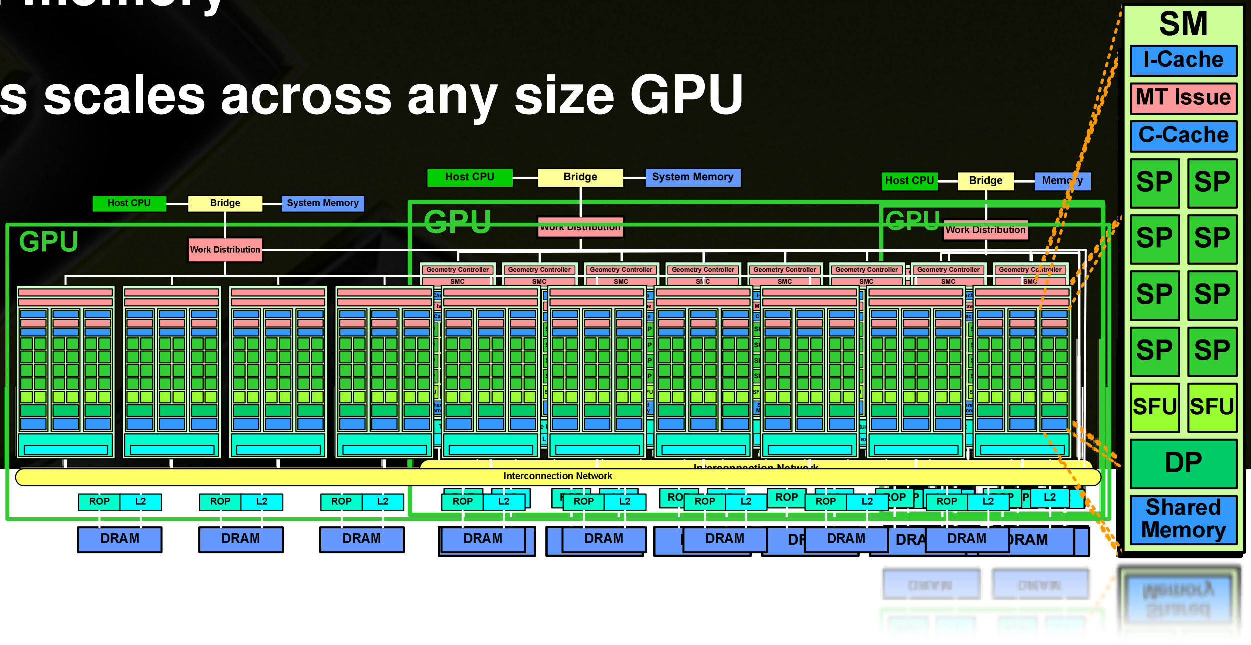
The Right Processor for the Right Tasks



NVIDIA CUDA Parallel Computing Architecture

- Many processors – eventually thousands
- Latency tolerant - execute 1000’s of threads
- General load/store
- On-chip shared-memory
- CUDA programs scales across any size GPU

240 SP Cores



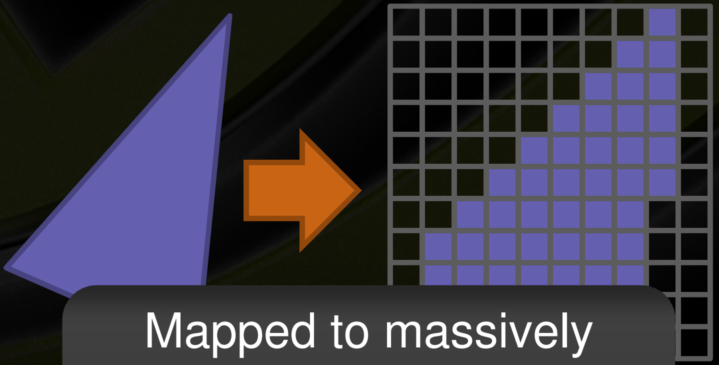
Rasterization & Ray Tracing

Rasterization

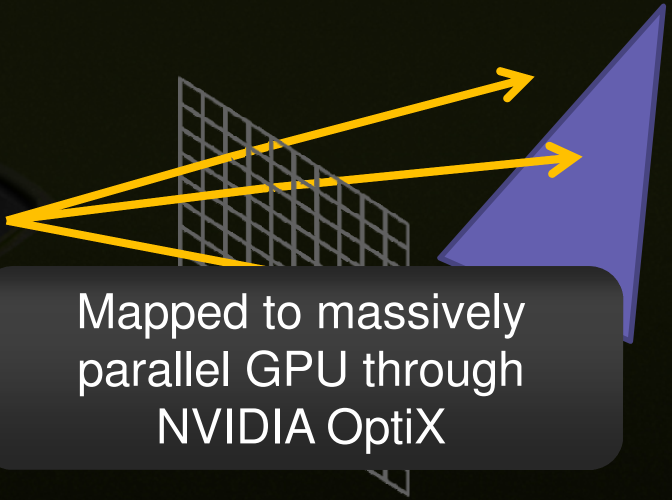
- For each triangle
  - Find the pixels it covers
  - For each pixel: compare to closest triangle so far

Classical Ray Tracing

- For each pixel
  - Find the triangles that might be closest
  - For each triangle: compute distance to pixel



Mapped to massively parallel GPU through DirectX or OpenGL



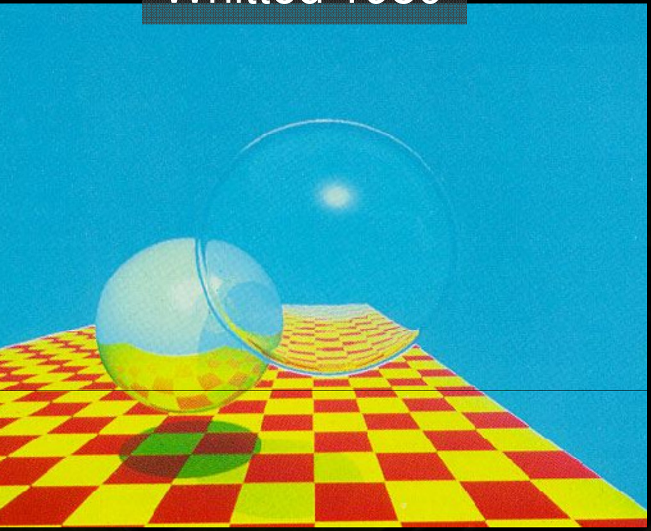
Mapped to massively parallel GPU through NVIDIA OptiX

Why ray tracing?

- Ray tracing unifies rendering of visual phenomena
  - fewer algorithms with fewer interactions between algorithms
- Easier to combine advanced visual effects robustly
  - soft shadows
  - subsurface scattering
  - indirect illumination
  - transparency
  - reflective & glossy surfaces
  - depth of field
  - …
- But: resource intensive, challenging to make fast

Ray tracing regimes

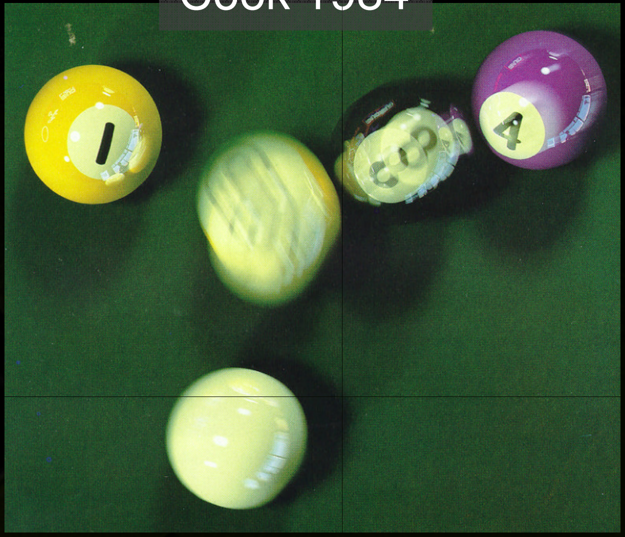
Whitted 1980



- Mirror reflections
- Perfect refractions
- Hard shadows

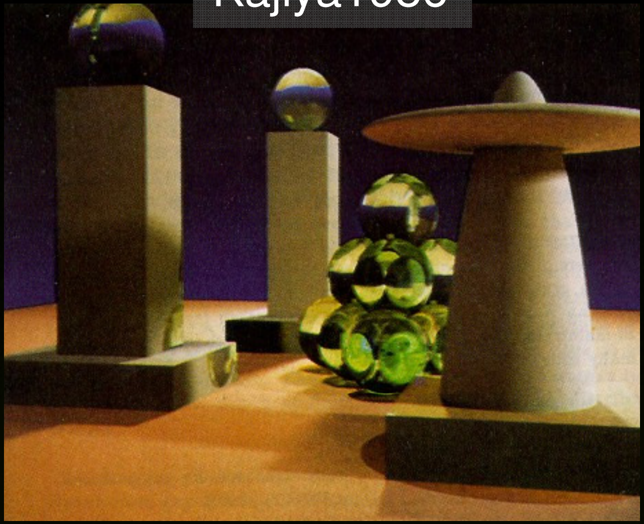
- 2-20 rays per pixel

Cook 1984



- Depth of field
- Motion blur
- Soft shadows
- Glossy reflections
- 20-200 rays per pixel

Kajiya1986



- Indirect illumination
- Caustics
- Physical accuracy

- 200-10^5 rays per pixel

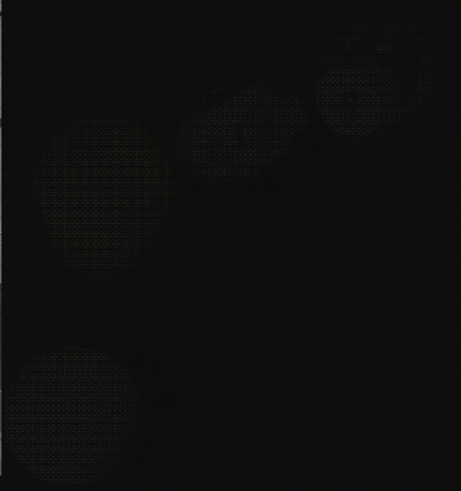
OptiX Examples

Interactive

•Mirror reflections
•Perfect refractions
•Hard shadows

•1-10 rays per pixel

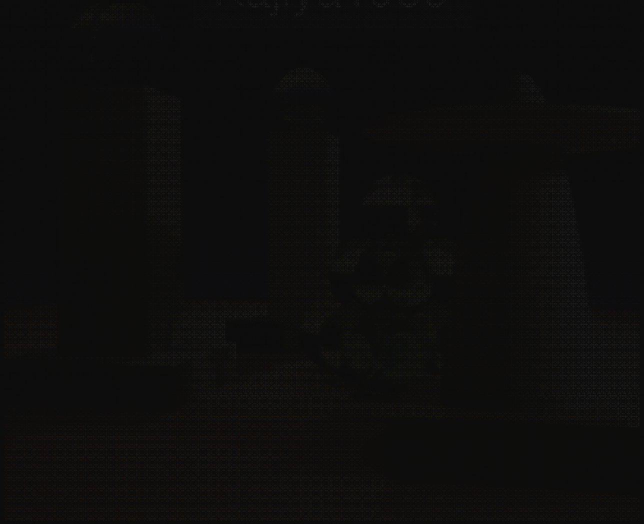
Cook 1984



•Depth of field
•Motion blur
•Soft shadows
•Glossy reflections

•10-100 rays per pixel

Kajiya1986

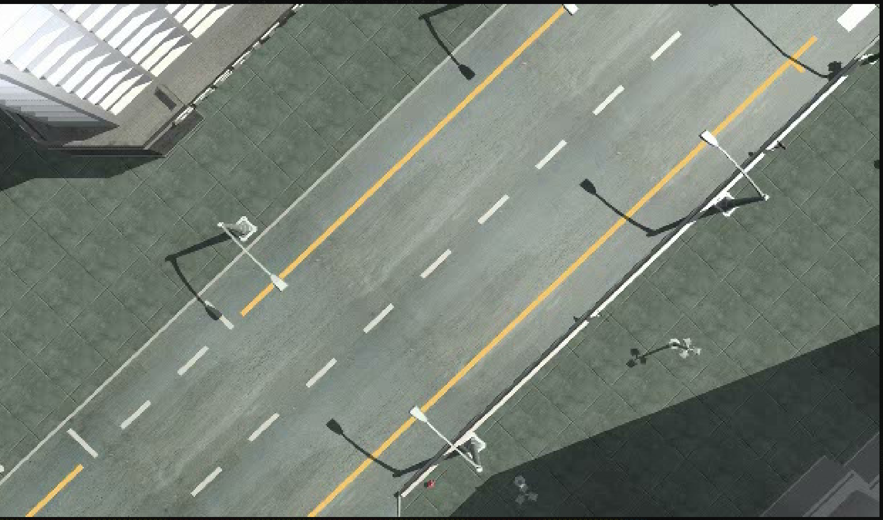


•Indirect illumination
•Caustics
•Physical accuracy

•100-10^5 rays per pixel

OptiX Examples

Cook 1984



Interactive

Progressive

- Mirror reflections
- Perfect refractions
- Hard shadows
- 1-10 rays per pixel

- Depth of field
- Motion blur
- Soft shadows
- Glossy reflections
- 10-100 rays per pixel

- Indirect illumination
- Caustics
- Physical accuracy
- 100-10^5 rays per pixel

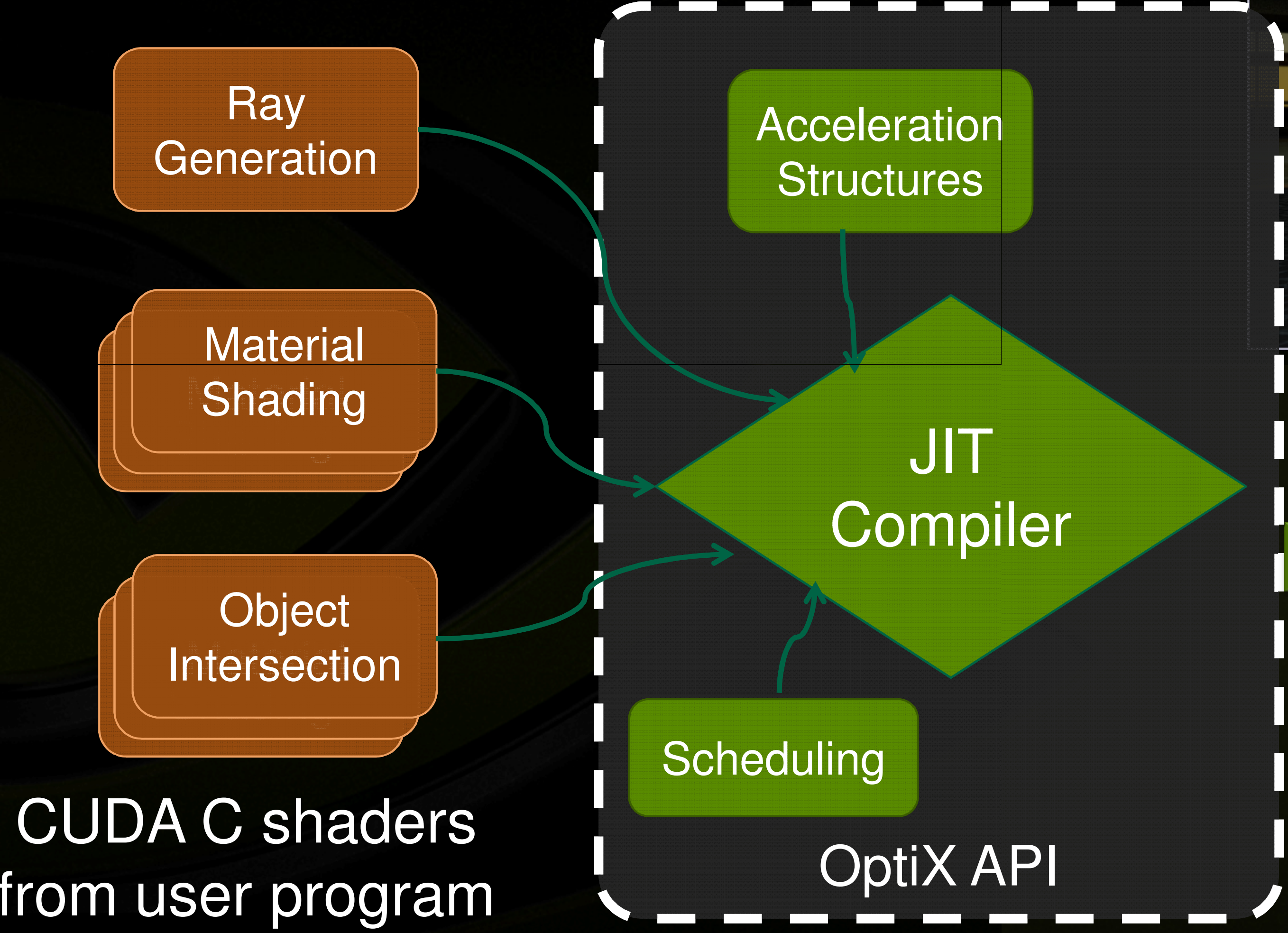
Programmable Operations

Rasterization	Ray Tracing (OptiX)
• Fragment	• Closest Hit • Any Hit
• Vertex • Geometry	• Intersection • Selector
	• Ray Generation • Miss • Exception

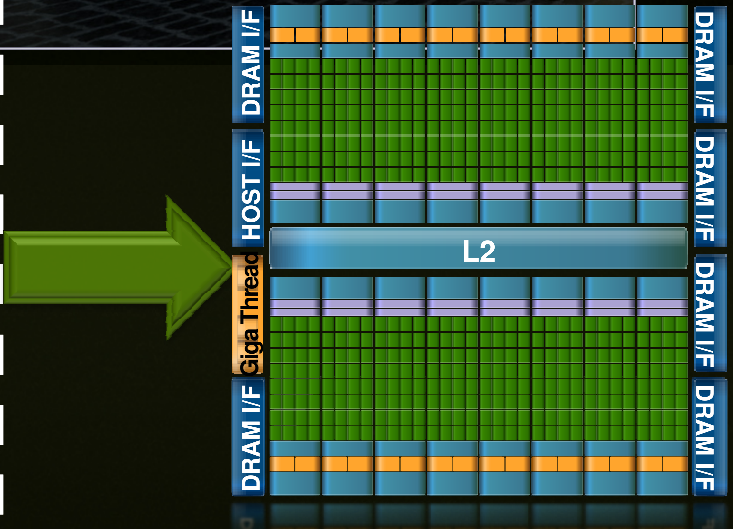
The ensemble of programs defines the rendering algorithm

(or collision detection algorithm, or sound propagation algorithm, etc.)

OptiX Functional Overview



CUDA C shaders from user program



GPU Execution via CUDA

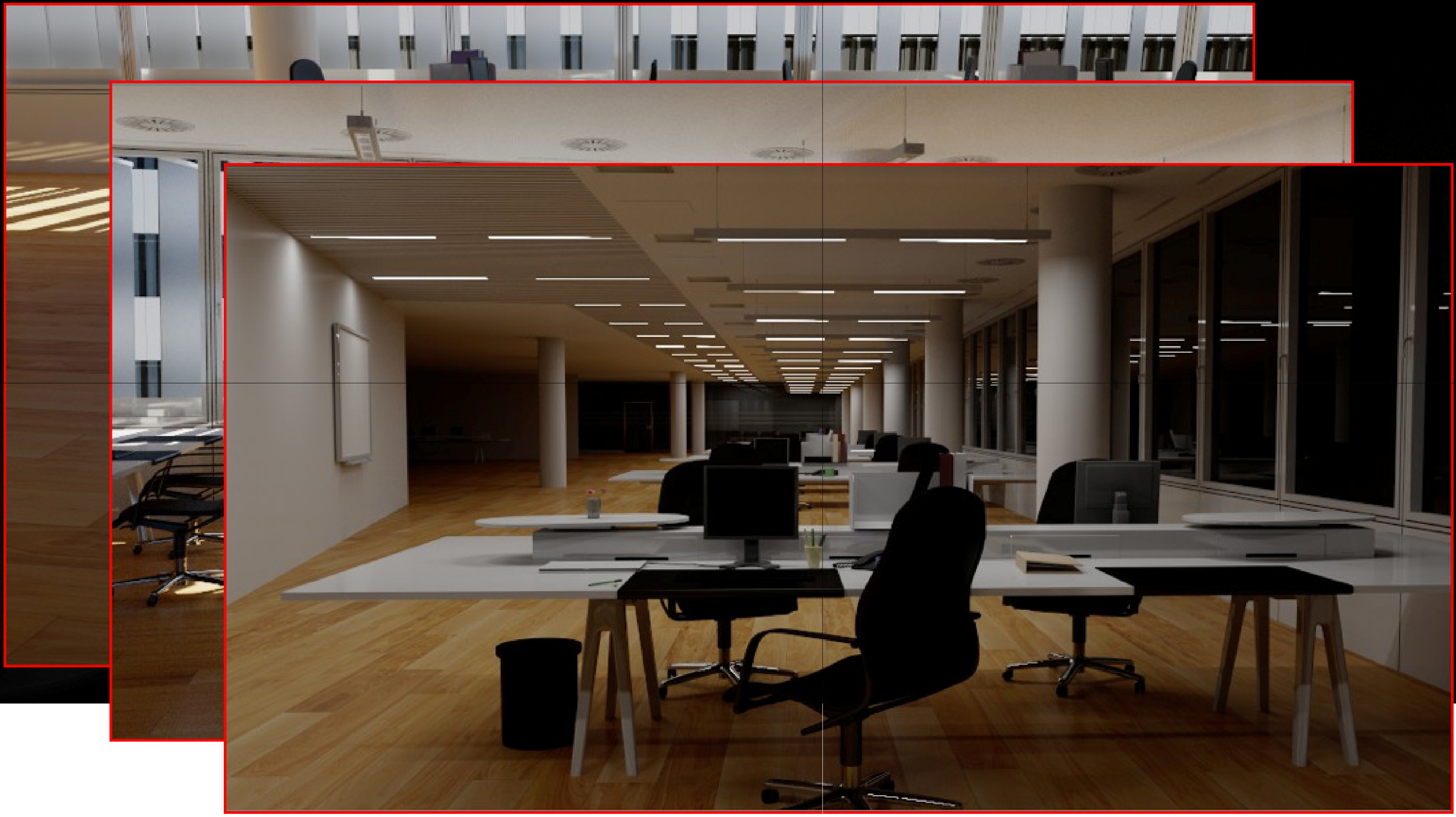
OptiX SDK

Available now at http://www.nvidia.com

iray® Light Transport Simulation

- No more renderer introduced artifacts
- No more parameter tweaking
- Deterministic quasi-Monte Carlo integro-approximation
    - consistent is faster than unbiased
    - converges unconditionally
    - exactly reproducible independent of parallelization
- High precision ray tracing
- Scales across architectures
    - takes optimal advantage of GPU compute power

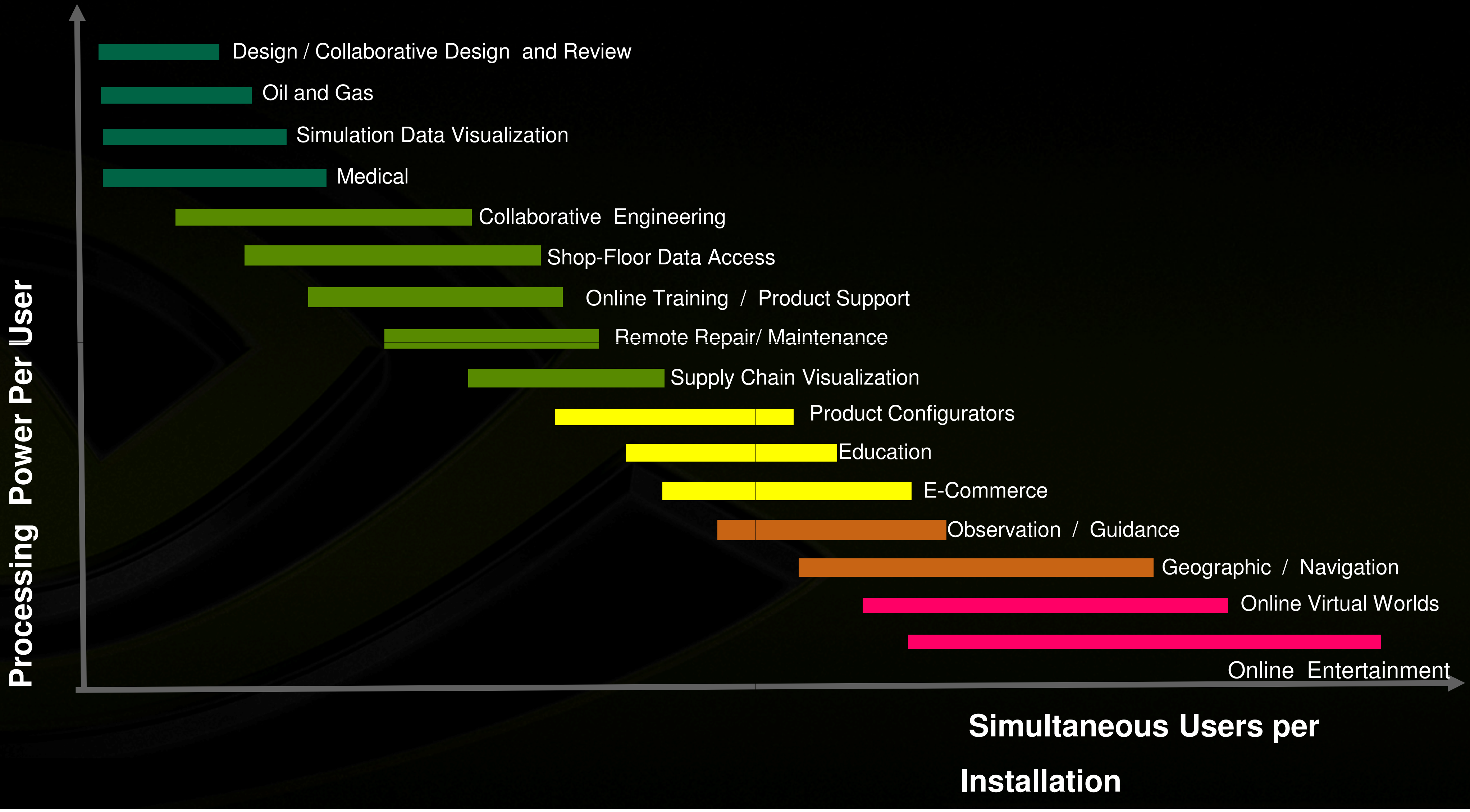
iray® Light Transport Simulation



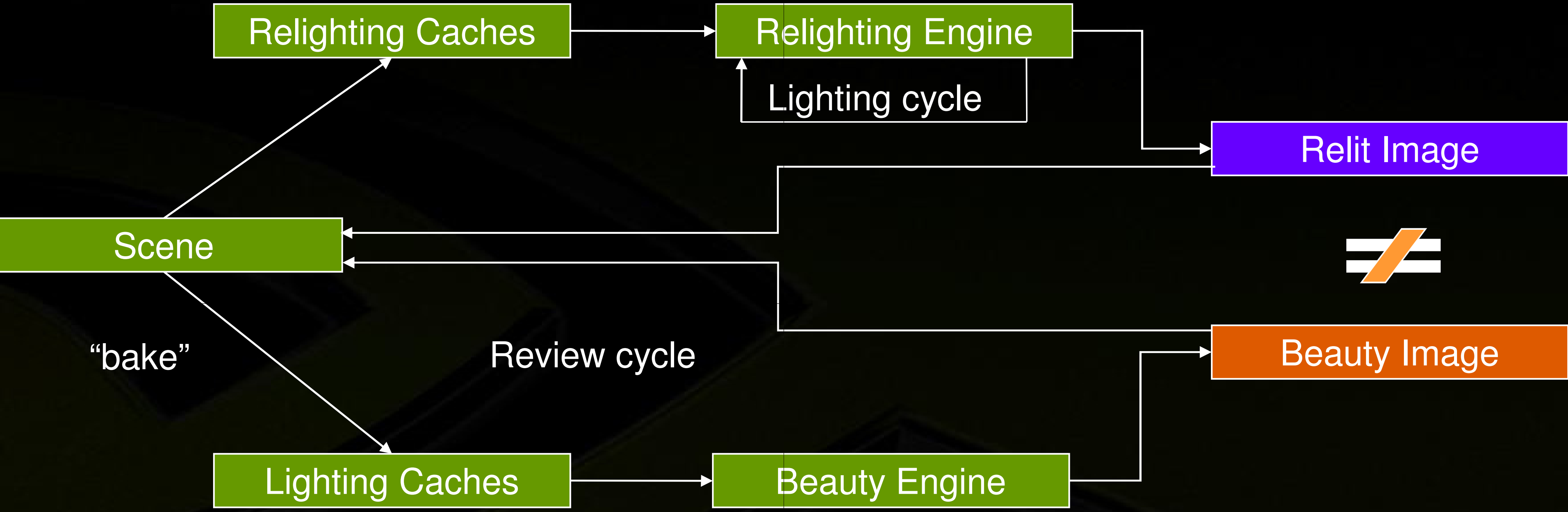
RealityServer

- Software Platform for 3D Applications and Web Services
  - remote interaction regardless of 3D data complexity
  - thin clients including mobile devices
  - collaboration
  - data security
  - built on top of Distributed Computing Environment (DiCE)
  - includes iray® GPU simulation technology

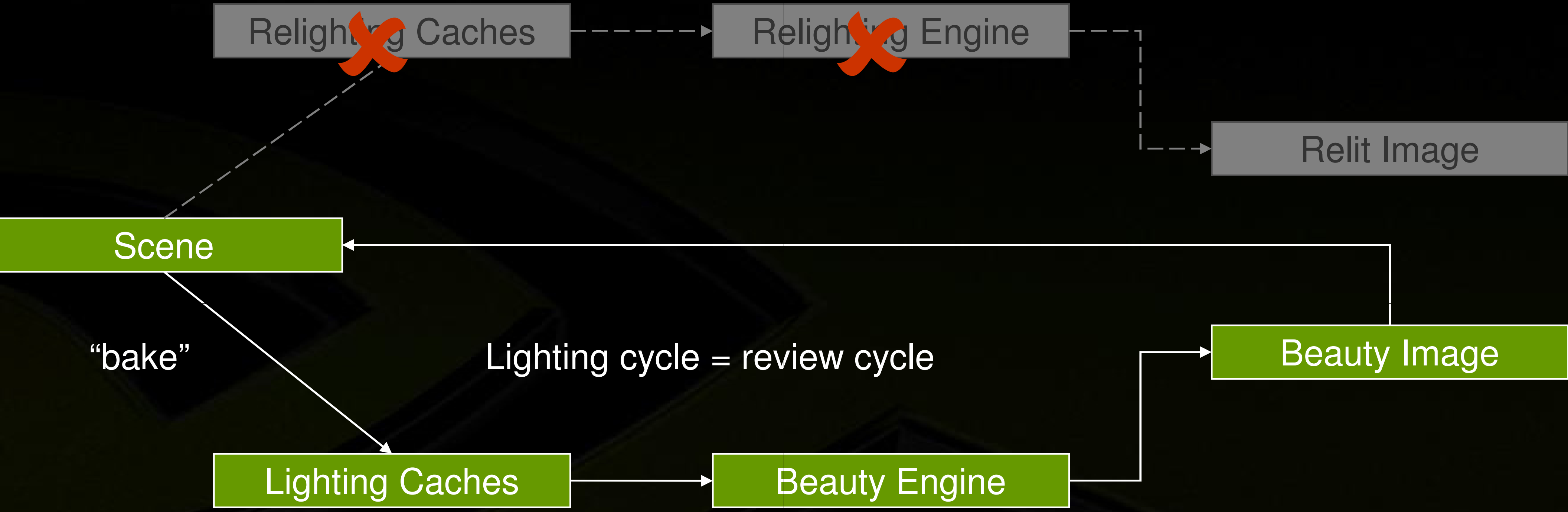
RealityServer Applications



A Typical Lighting Cycle



Beauty lighting



PantaRay – Precomputation Engine

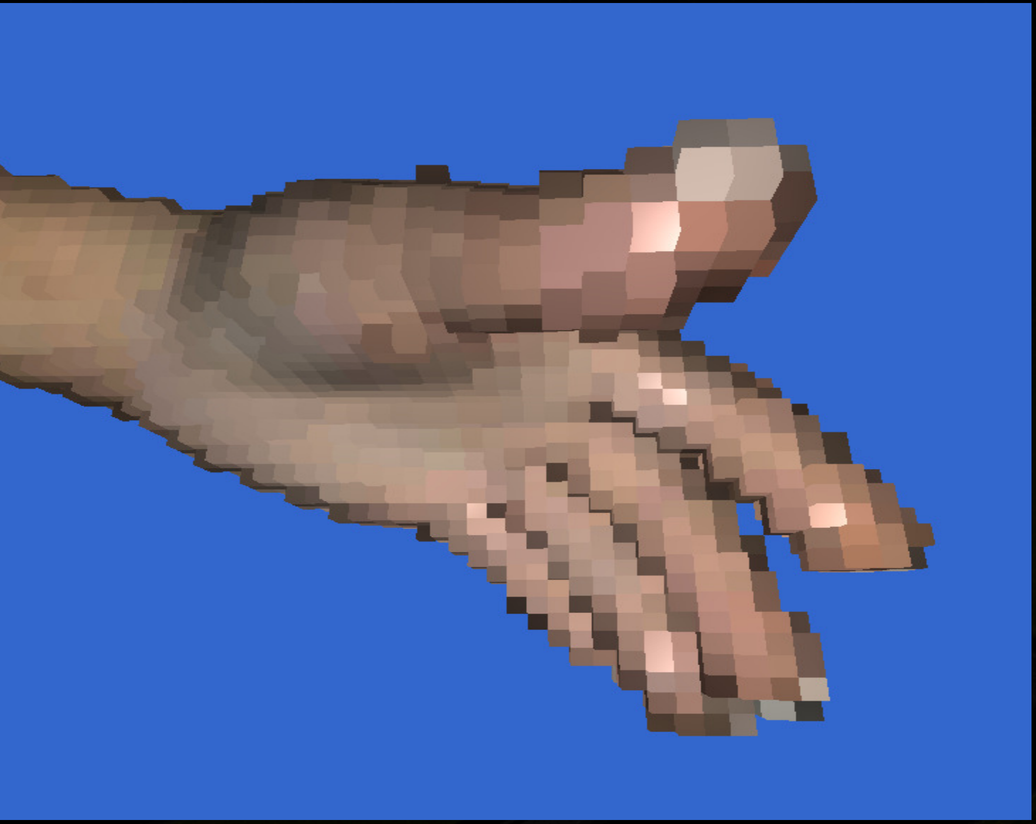
- Special purpose caching engine based on data structures optimized for GPU deployment
- Extreme scene complexity
  - Normal shots in the 10-100M polygons
  - Complex shots over 1G polygons
- Accelerate beauty pass with raytraced PRT caches
- Compared to the previous method, the tractability limit is increased by 2 orders of magnitude (100x)

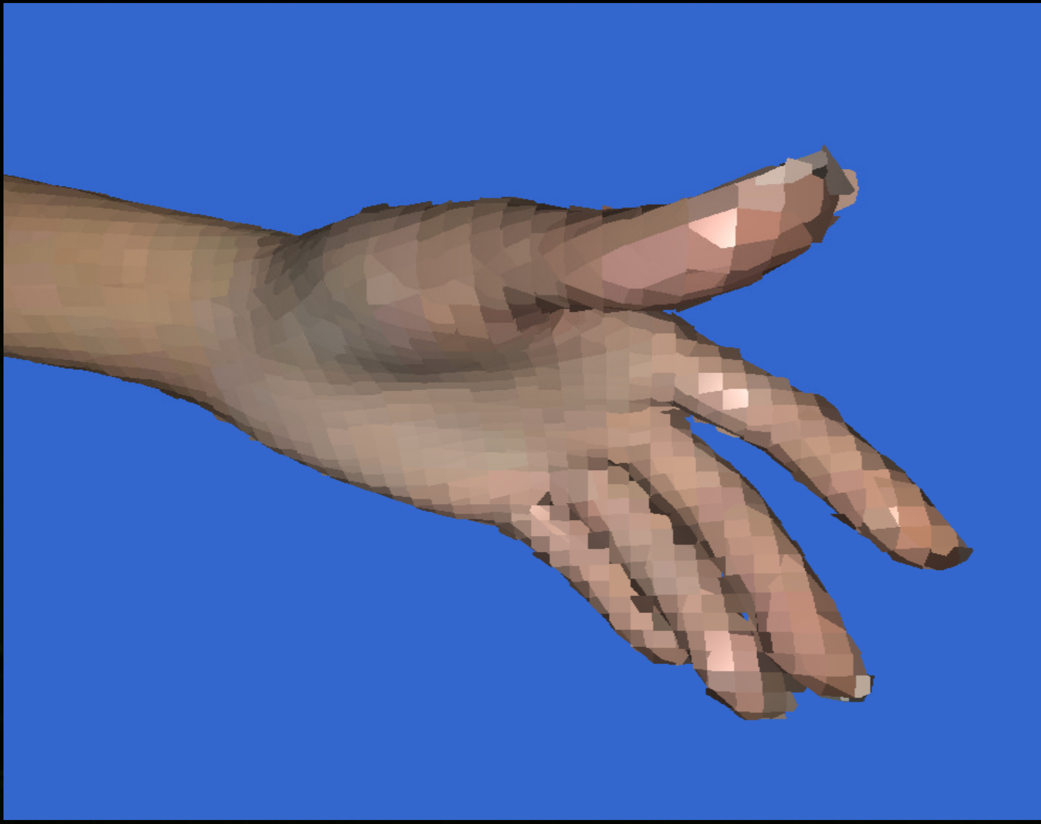
Voxel Rendering: Sibenik

Voxel Rendering: Why Voxels?

- Future challenges
  - More geometric detail
  - Unique assets
  - Authoring, capturing
  - Ray tracing
- Hot Idea?
  - Leading game developers looking at voxels
  - Web press, game technology fans talking it up
- Historical trends
  - Raw data wins
  - Simplicity turns into efficiency – think of Z-buffer!

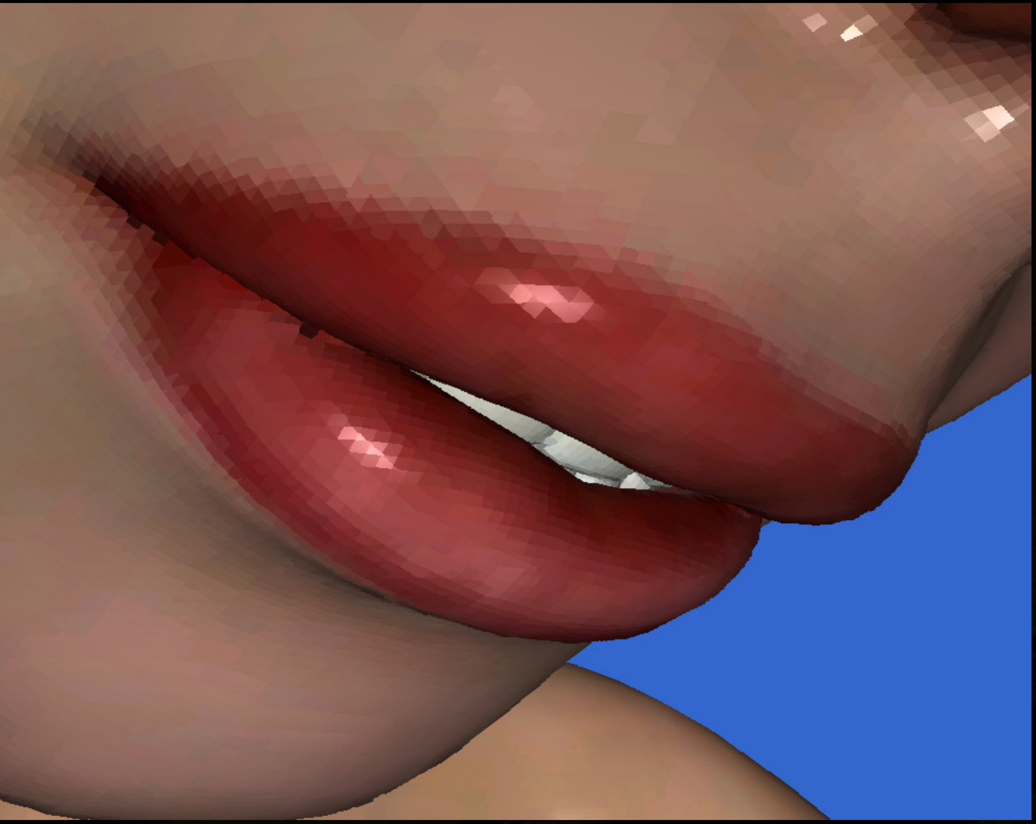
Contours

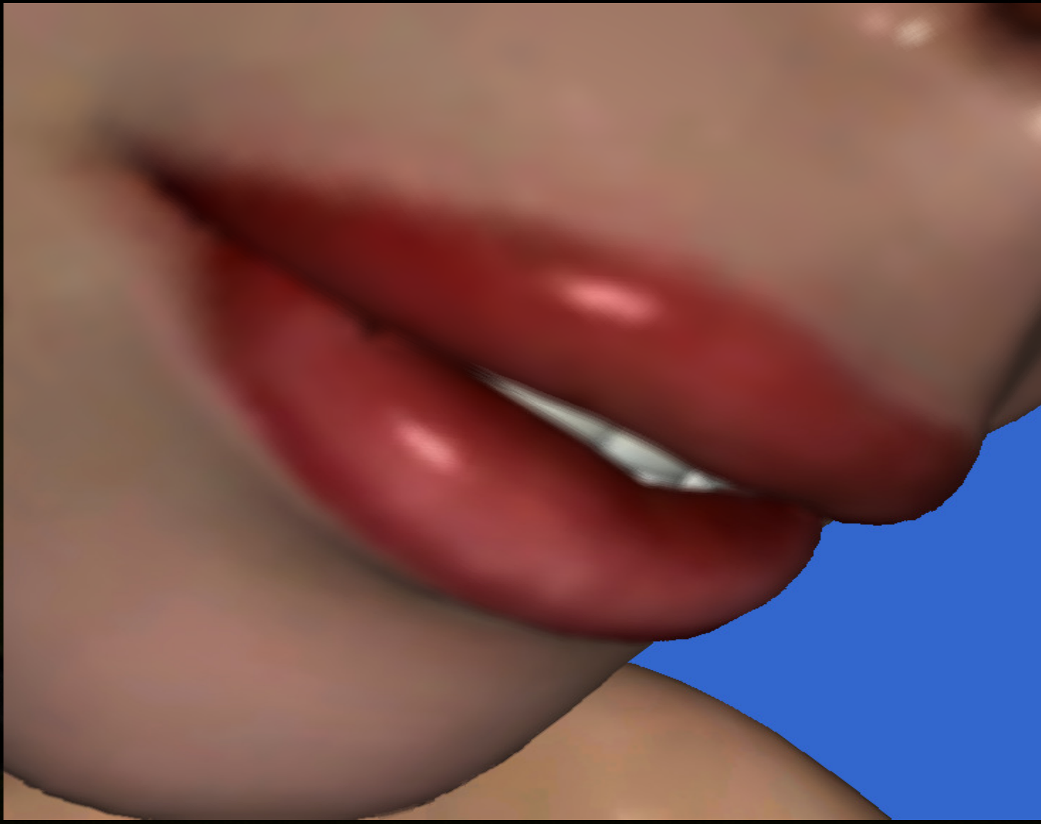




ote: low-resolution voxelization

Post-Process Blur

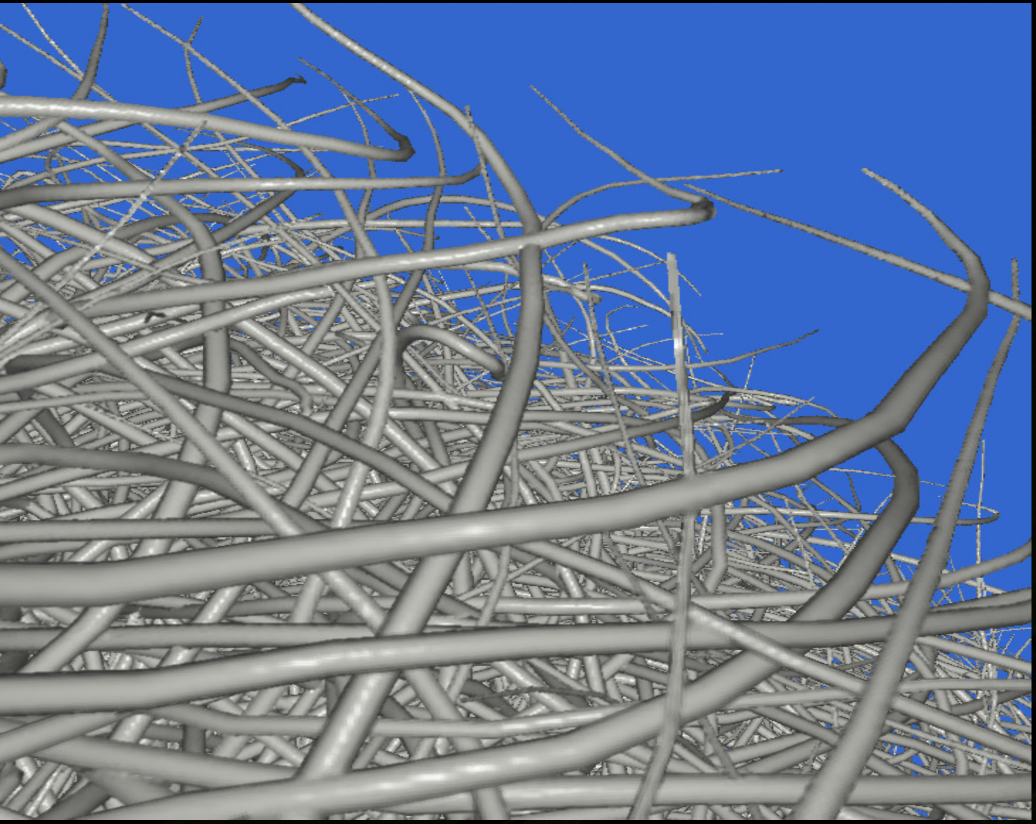

No blur

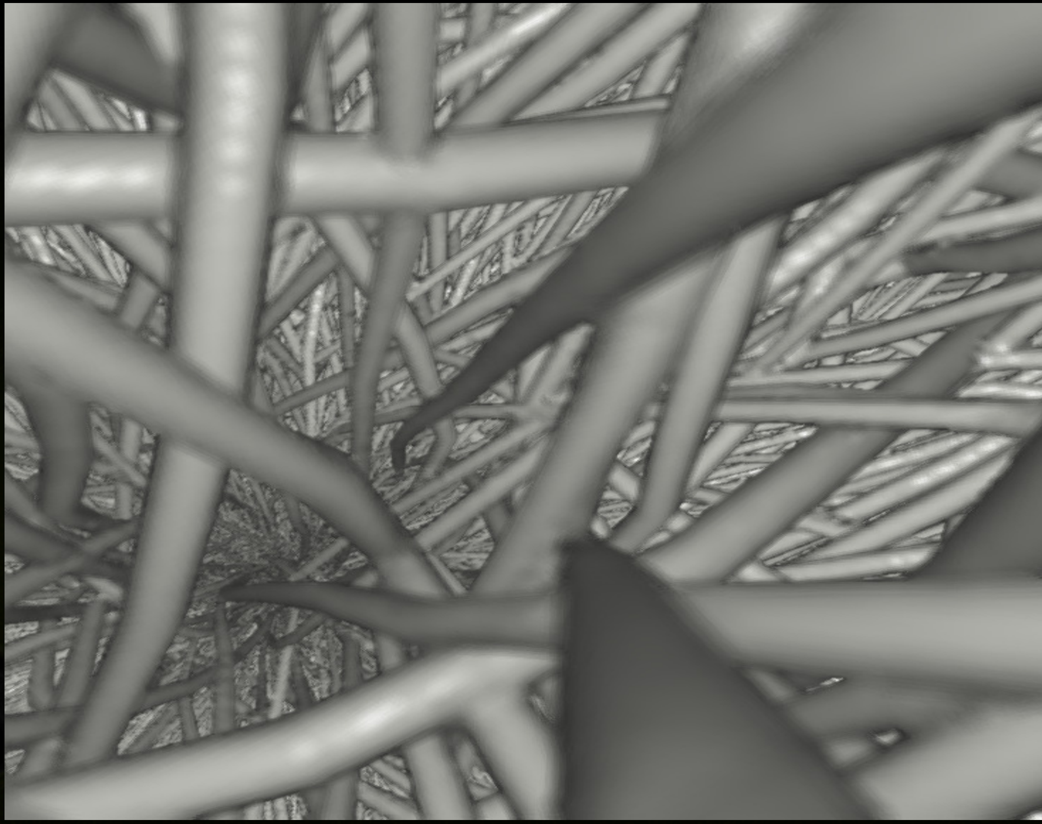

With blur

ote: low-resolution voxelization

Hairball, 11 levels: 10:28, 1552 MB, 28.4 Mrays/s

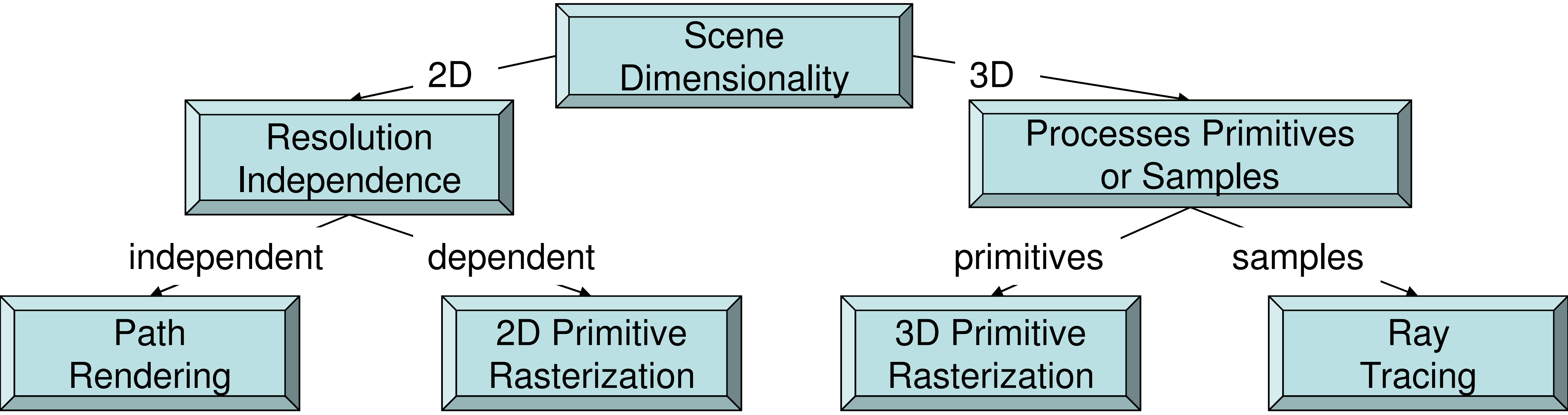
Resolution Highlights





Hairball, 11 levels: 1552 MB

Taxonomy of Rendering



Examples
PostScript, PDF, SVG, TrueType, Adobe Flash, Microsoft Silverlight, Apple Quartz 2D

Examples
GDI, Xlib

Examples
OpenGL, Direct3D

Examples
Mental Ray
iRay
Optix

Conventional wisdom says GPUs don't accelerate these rendering paradigms

As usual, the conventional wisdom is WRONG ☺

Path Rendering Standards

Document Printing and Exchange

- PDF
- Adobe PostScript
- *Open XML Paper (XPS)*

Resolution-Independent Fonts

- *OpenType*
- *TrueType*

Immersive Web Experience

- *Flash*
- Microsoft Silverlight
- SVG *Scalable Vector Graphics*
- *HTML 5*

2D Graphics Programming Interfaces

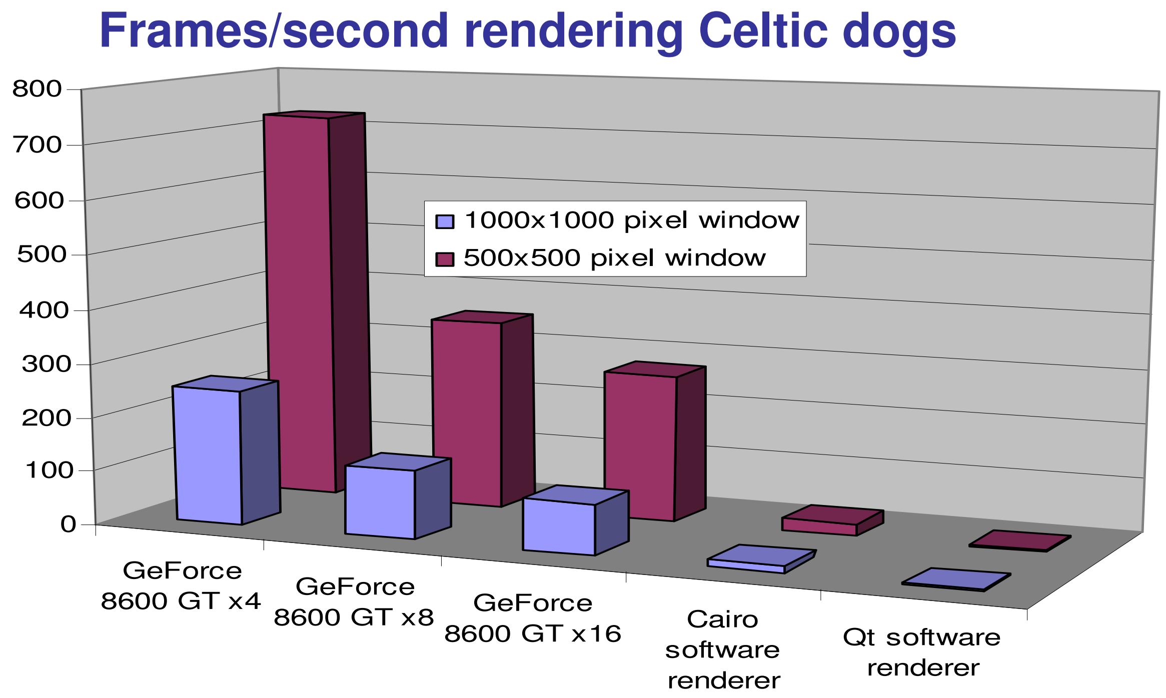
- Java *Java 2D API*
- Qt *QtGui API*
- Quartz 2D Graphics *Mac OS X 2D API*
- OpenVG *Khronos API*

Office Productivity Applications

- Microsoft Office
- OpenOffice.org
- Ai *Adobe Illustrator*
- *Open Source Inkscape*

GPU vs. CPU Path Rendering Performance

“Celtic round dogs” scene
1 very complex path
5,031 cubic Bezier commands
29,068 coordinates



	GeForce 8600 GT x4	GeForce 8600 GT x8	GeForce 8600 GT x16	Cairo software renderer	Qt software renderer
🟦 1000x1000 pixel window	250	126	93	13.1	1.1
🟥 500x500 pixel window	725	350	272	22.2	2.7

GPU-Accelerated Path Rendering Exploits GPU Advantages

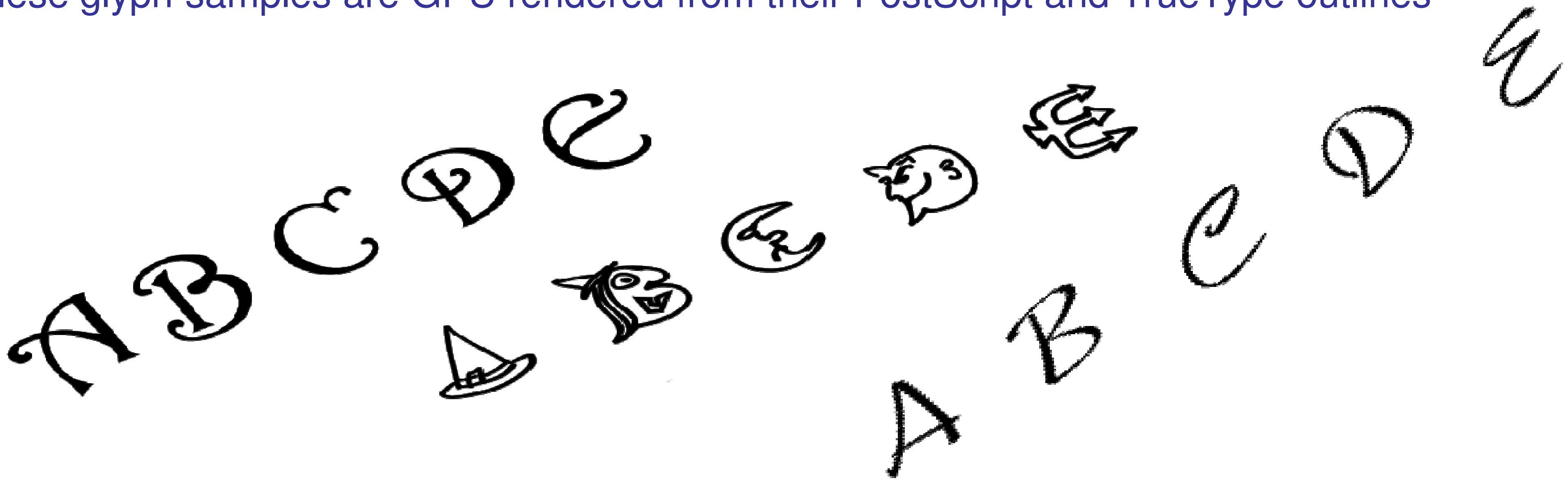
- Use stencil to track filled and stroked path coverage
  - GPUs rasterize stencil only at 2x to 3x faster than normal peak shaded rendering rate
  - [Loop & Blinn 2005] discard shader techniques support rasterizing paths with curved segments
  - No tessellation required
    - Avoids expensive, sequential process
- Then shade path coverage with conventional GPU-accelerated shading
  - GPUs far better at shading than CPUs
- Antialiasing - Hardware multisampling very efficient

All path content below rendered by GPU <u>without</u> tessellation

Even Font Rendering by GPU

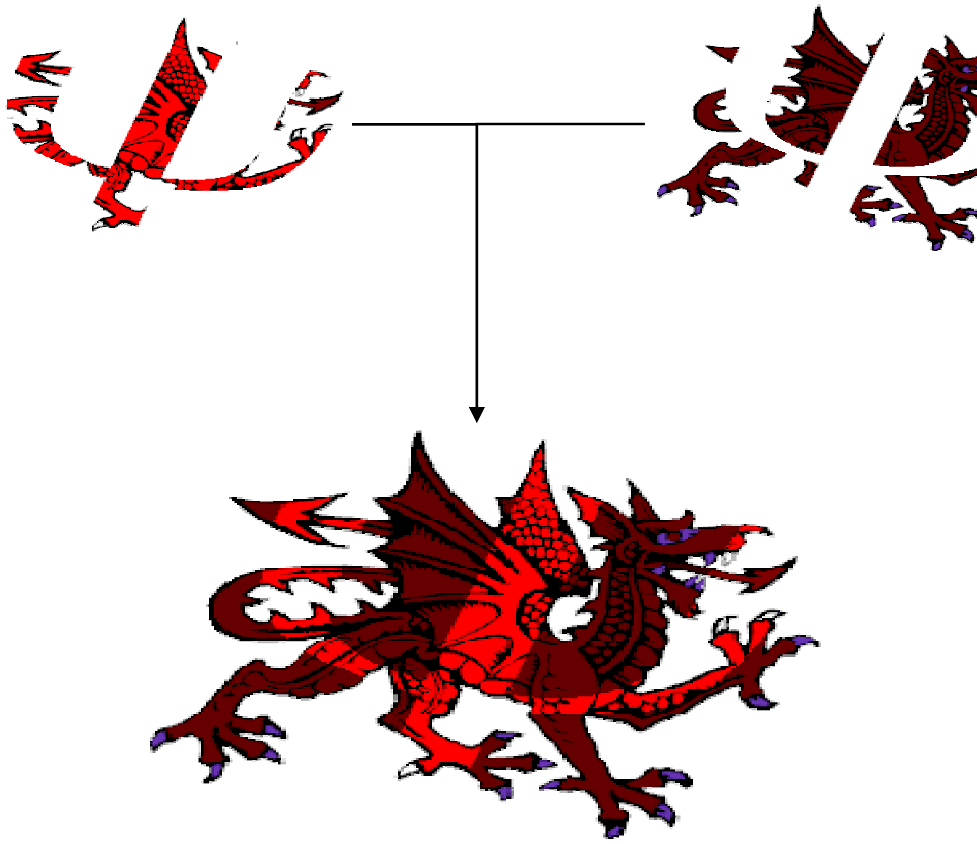
- GPU-accelerated font rendering means the end of glyphs rasterized to bitmaps
  - GPU can render directly from outlines

hese glyph samples are GPU rendered from their PostScript and TrueType outlines



GPU Advantage: Example of Clipping to an Arbitrary Path

- Path rendering standards require path clipping
  - A rendered path can be clipped to another arbitrary path
- Expensive on the CPU
  - Must compute intersection of two arbitrary paths
- Cheap on the GPU
  - Stencil buffering is very efficient


Dragon masked by letter Psi

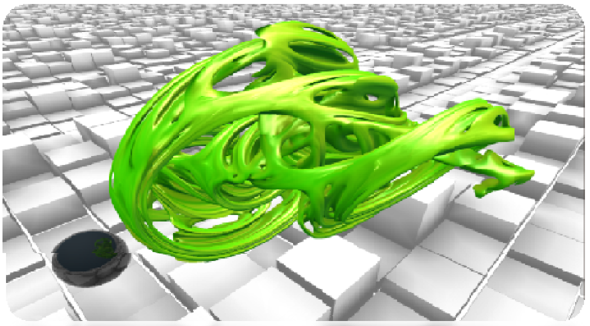
low Far Beyond?

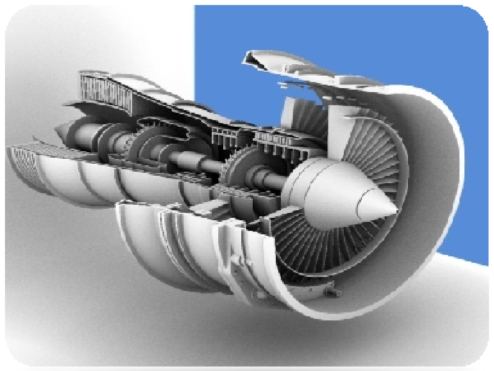
The computational graphics continuum

"Just" programmable shading: DX, OGL

low Far Beyond?

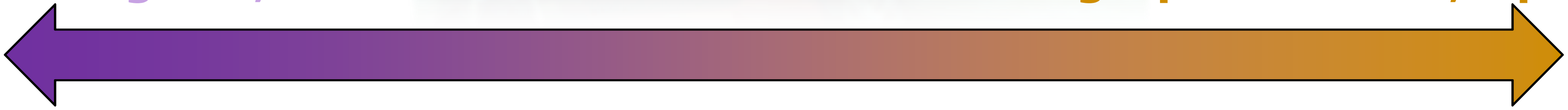
The computational graphics continuum





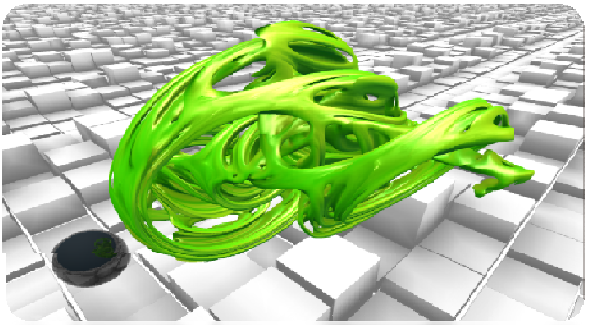
"Just" programmable shading: DX, OGL

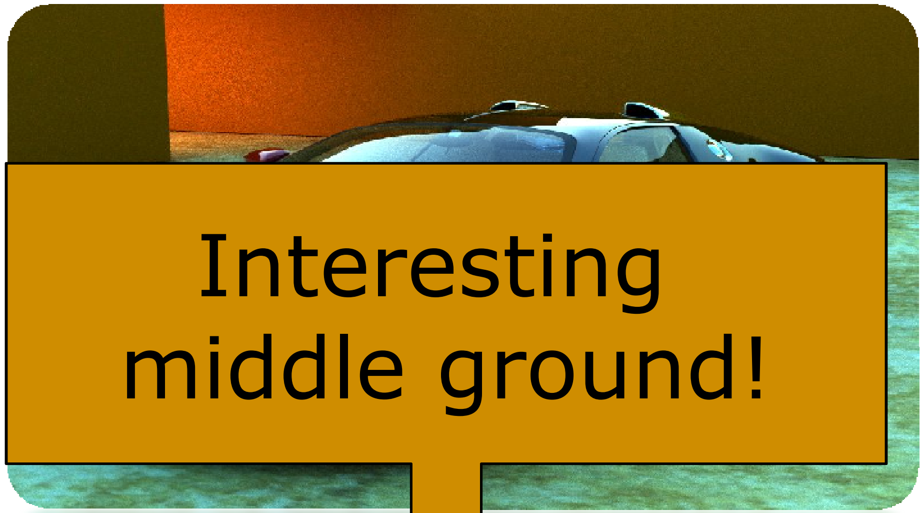
"Pure" compute-based graphics: CUDA, OptiX



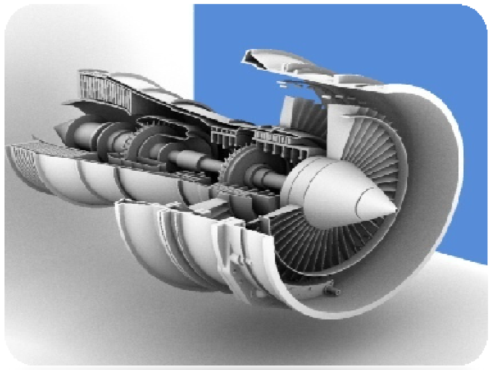
low Far Beyond?

The continuum "Beyond Programmable Shading"





Interesting middle ground!



"Just" programmable shading: DX, OGL

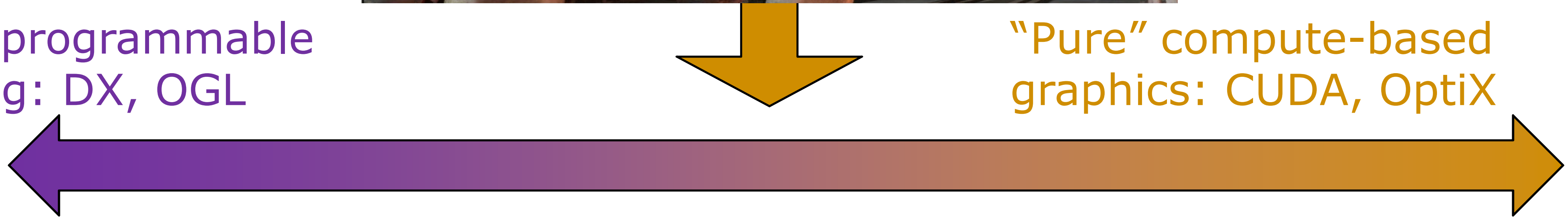
"Pure" compute-based graphics: CUDA, OptiX

low Far Beyond?

Image Space Photon Mapping

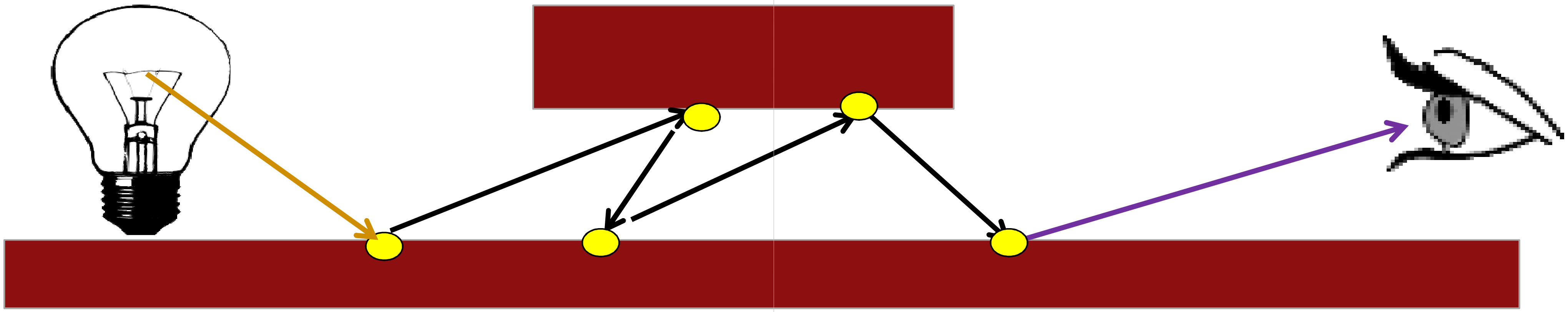
st” programmable
ading: DX, OGL

“Pure” compute-based
graphics: CUDA, OptiX



“Image-Space Photon Mapping”,
Morgan McGuire, David Luebke. High Performance Graphics 2009

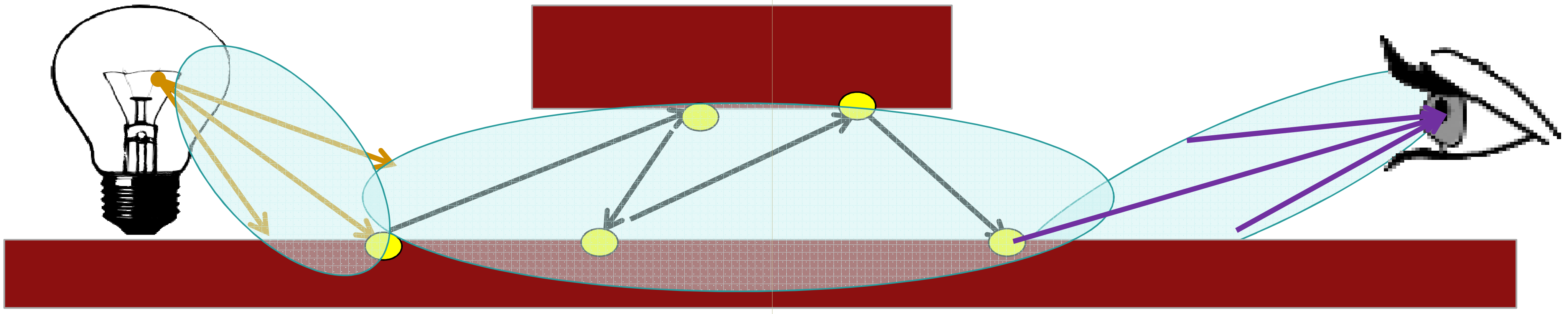
6



Slide 46

DPL16 I sped up the wipes to 0.2s, see what you think.
David Luebke, 7/31/2009

3



Photon Mapping Time (seconds)

e.g., [Jensen 01, Purcell et al. 03, Zhou et al. 08]

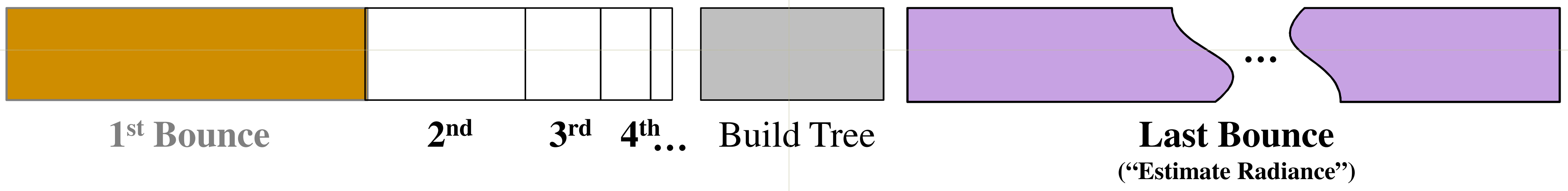
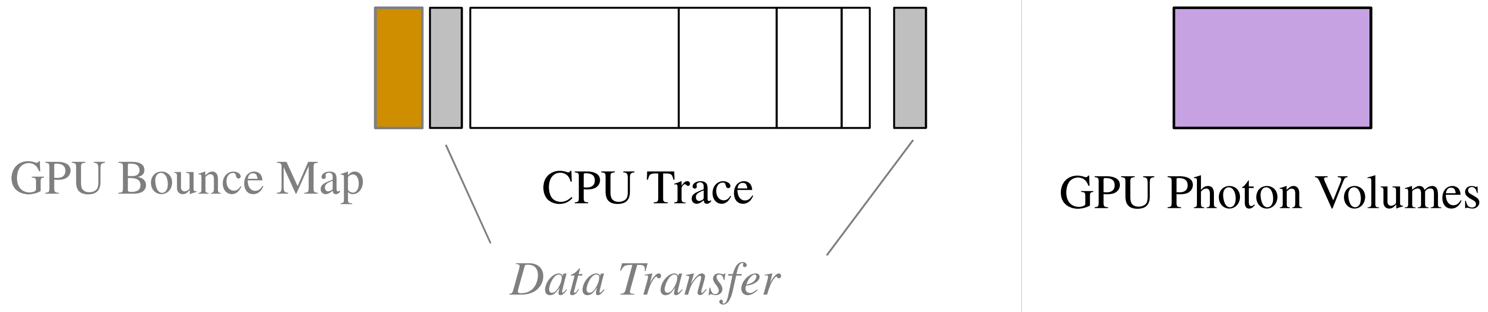
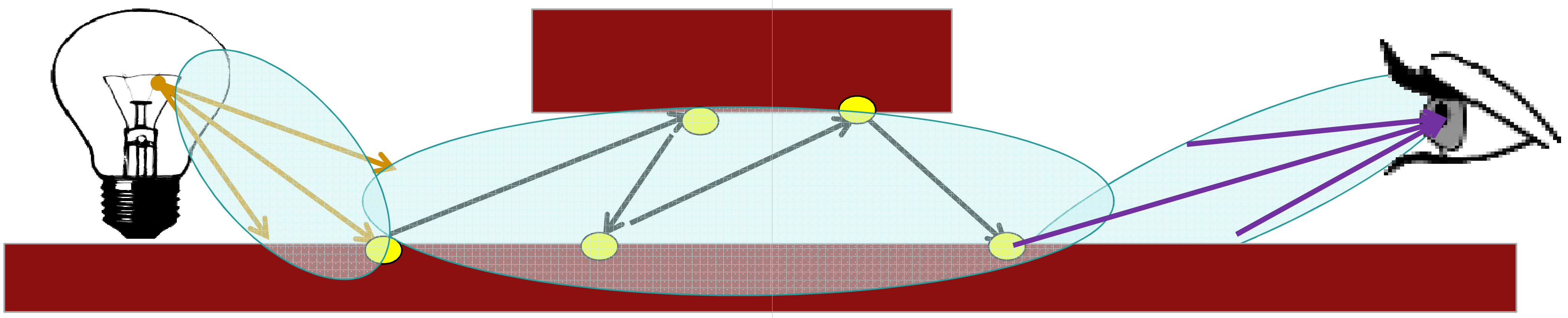


Image Space Photon Mapping (*mi*DPL17*onds*)



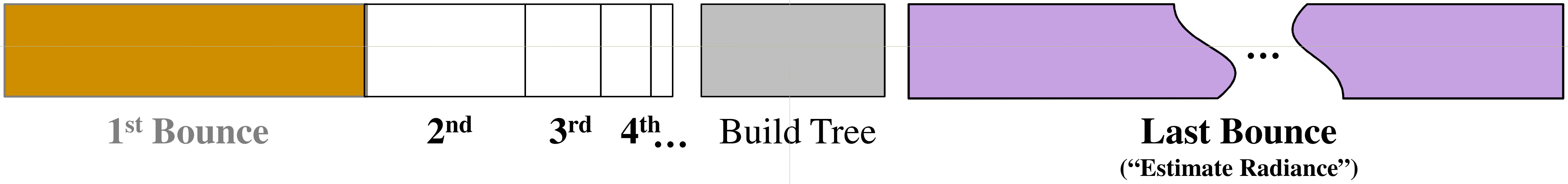
Slide 47

DPL17	technically its rebalancing the tree, right? doesn't traditional PM build the tree incrementally then balance it to speed up the knn-search? David Luebke, 7/31/2009
DPL18	I rearranged the arrows and animation a bit to emphasize the single point of projection for first and last bounces, and to make the "appear" animations into ultra-fast (0.2s) fades. David Luebke, 7/31/2009

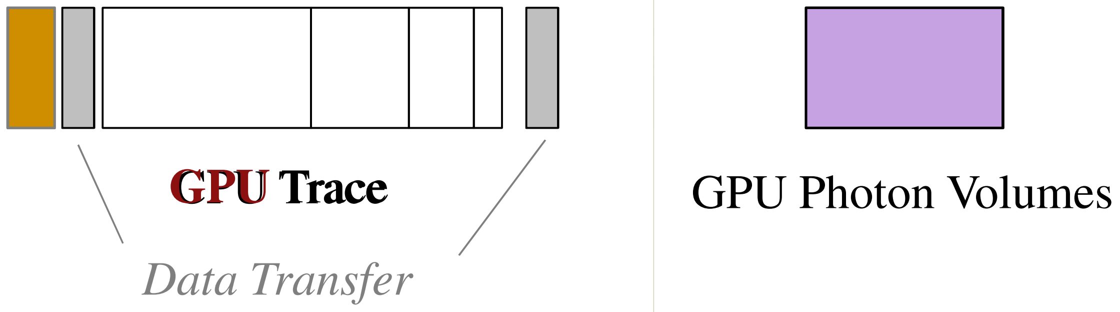


Photon Mapping Time (seconds)

e.g., [Jensen 01, Purcell et al. 03, Zhou et al. 08]



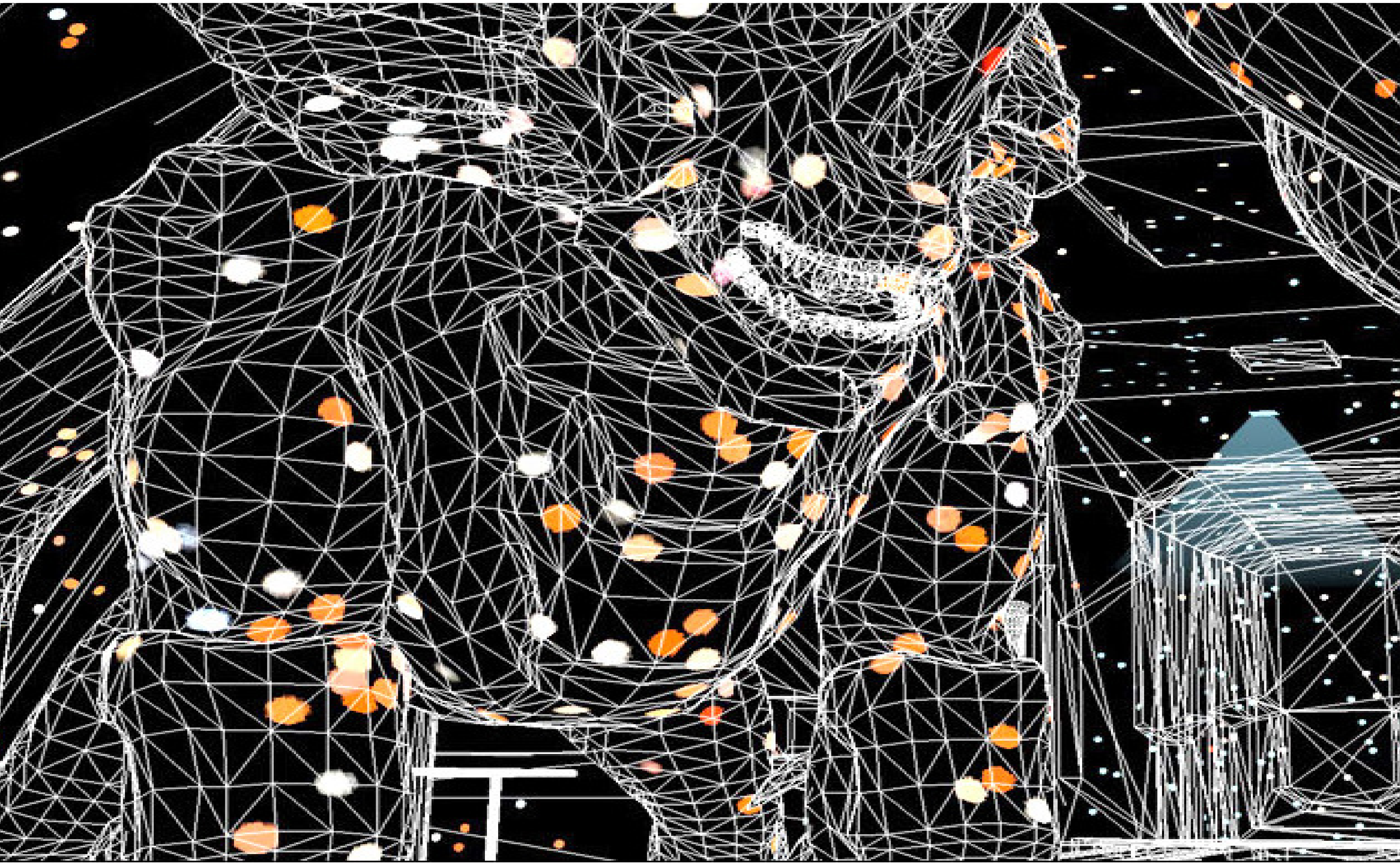
OptiX-enabled Image Space PhotoDPL19apping



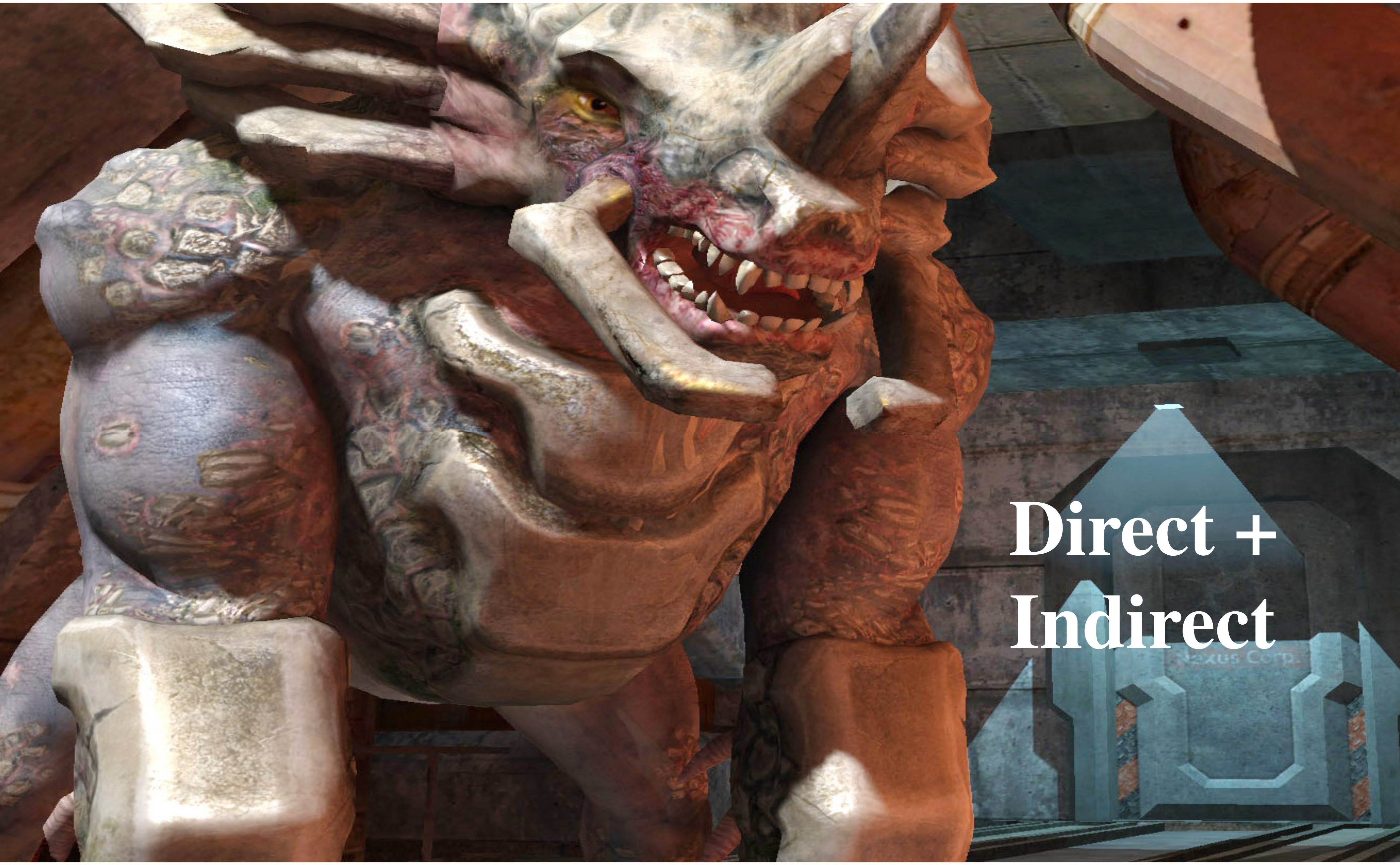
Slide 48

DPL19	technically its rebalancing the tree, right? doesn't traditional PM build the tree incrementally then balance it to speed up the knn-search? David Luebke, 7/31/2009
DPL20	I rearranged the arrows and animation a bit to emphasize the single point of projection for first and last bounces, and to make the "appear" animations into ultra-fast (0.2s) fades. David Luebke, 7/31/2009

Direct

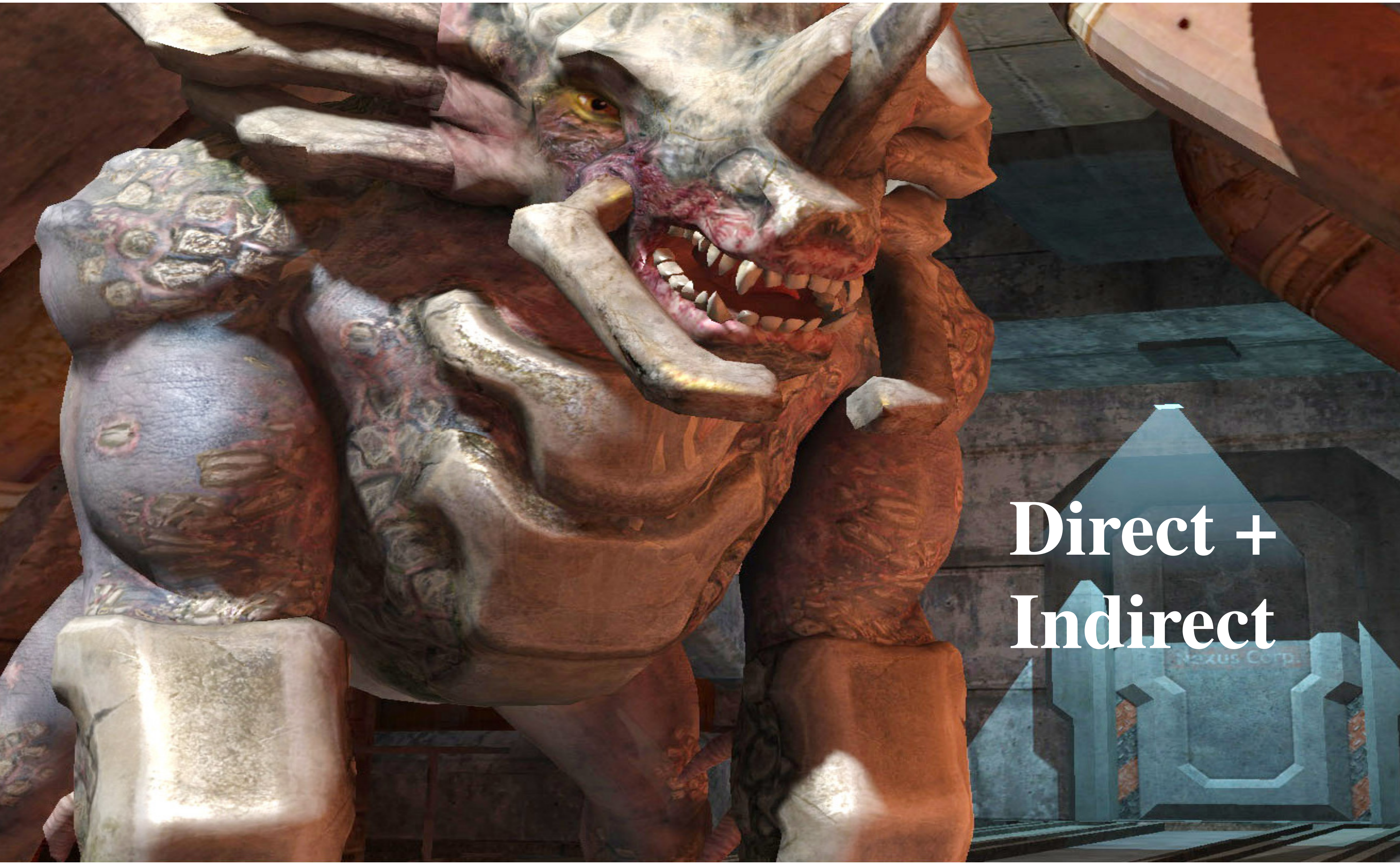


Indirect



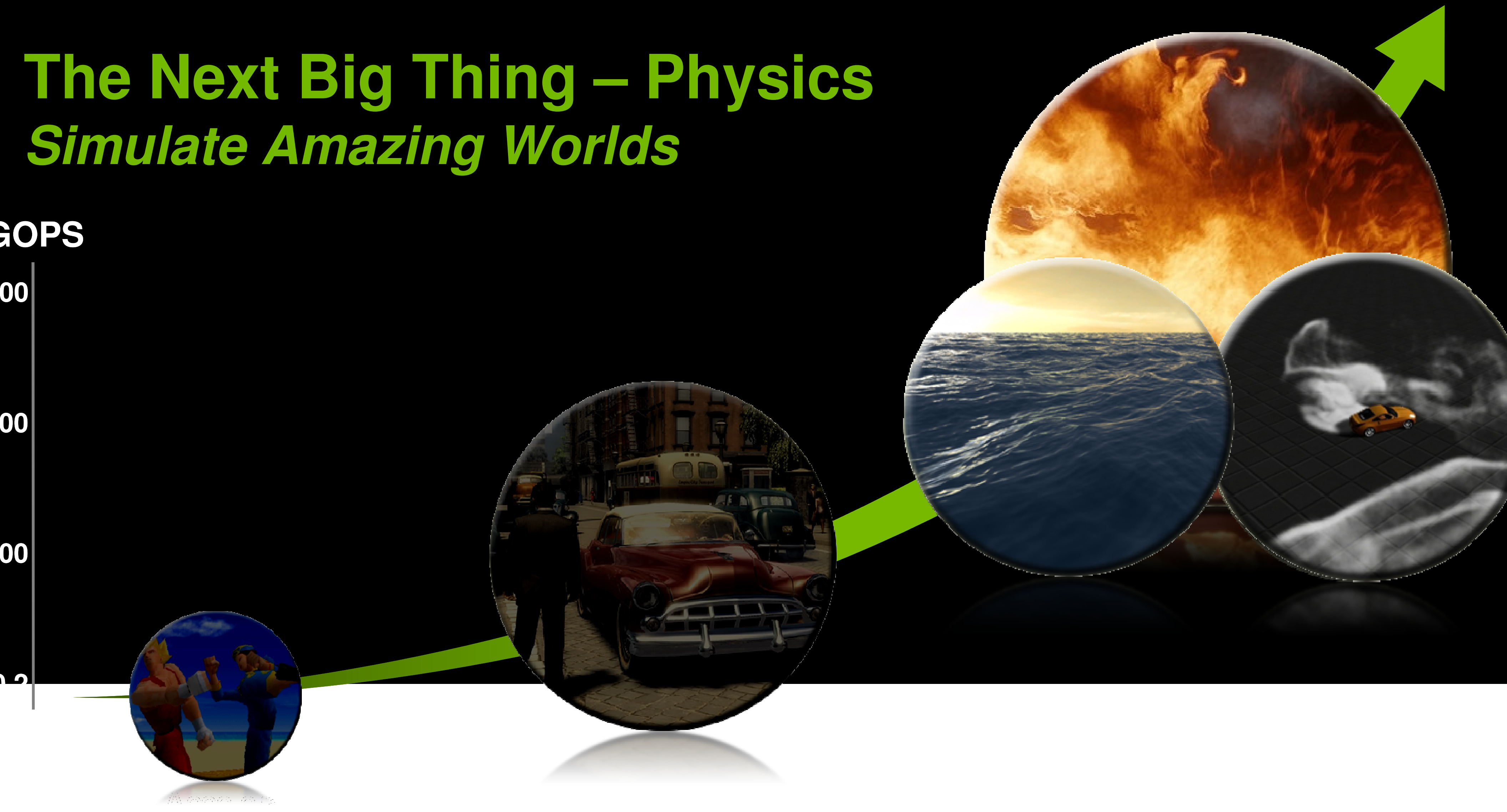
Direct +
Indirect

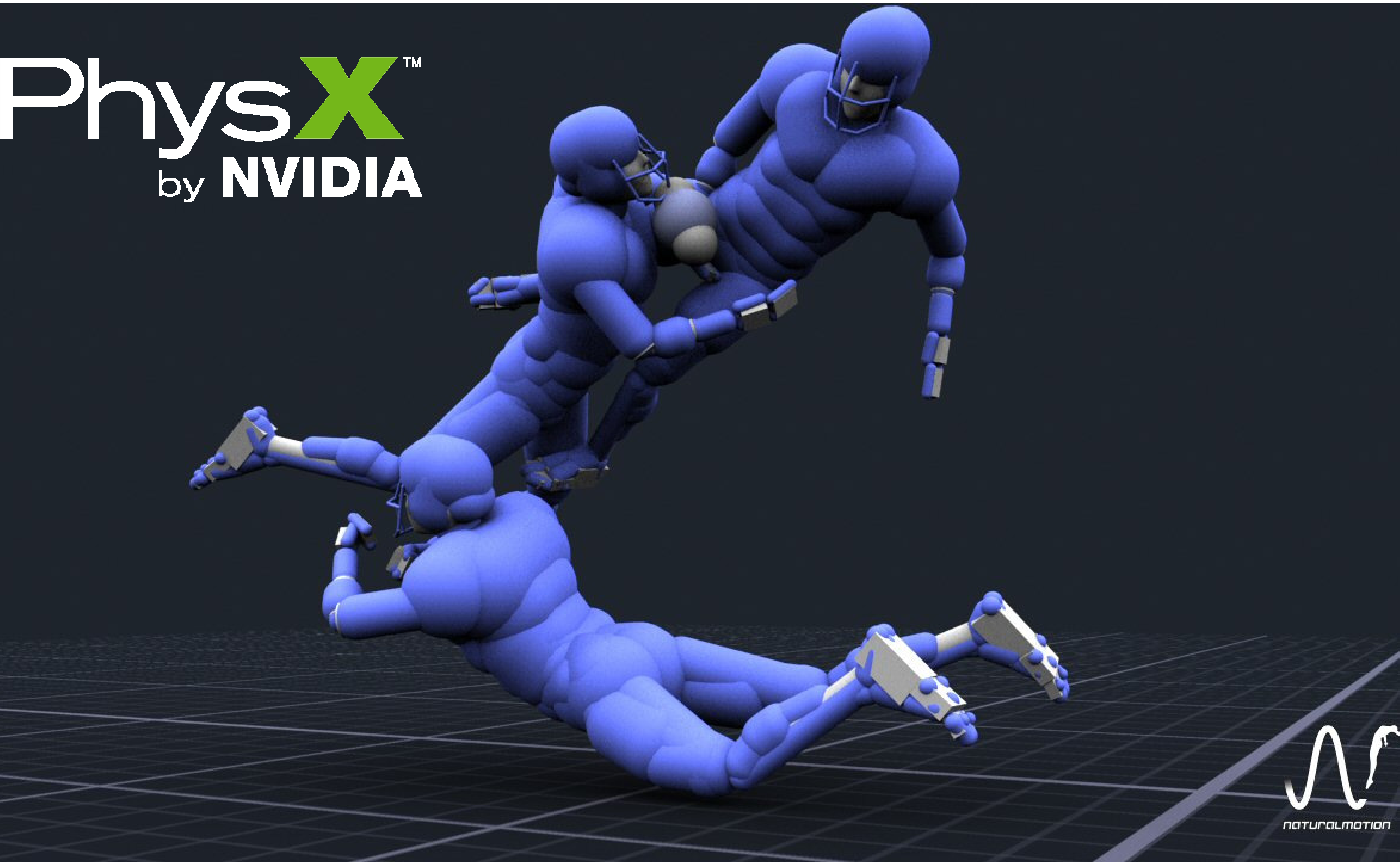
Direct +
Ambient


Direct +
Indirect
Nexus Corp.

The Next Big Thing – Physics

Simulate Amazing Worlds




PhysX™
by NVIDIA
naturalmotion

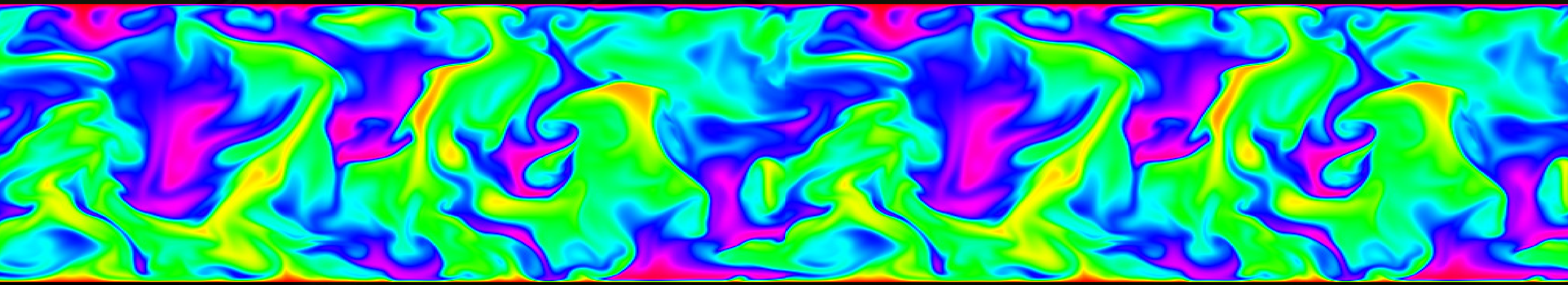
Real-time fluid effects

Complex fluid-drive motion is all around

- Car exhaust, dust storms, rolling mist, steam, smoke, fire, contrails, bubbles in water, …

Goal: Add this level of realism to games

Problem: Turbulent motion is computationally intensive!



Solution: GPUs are computational monsters!

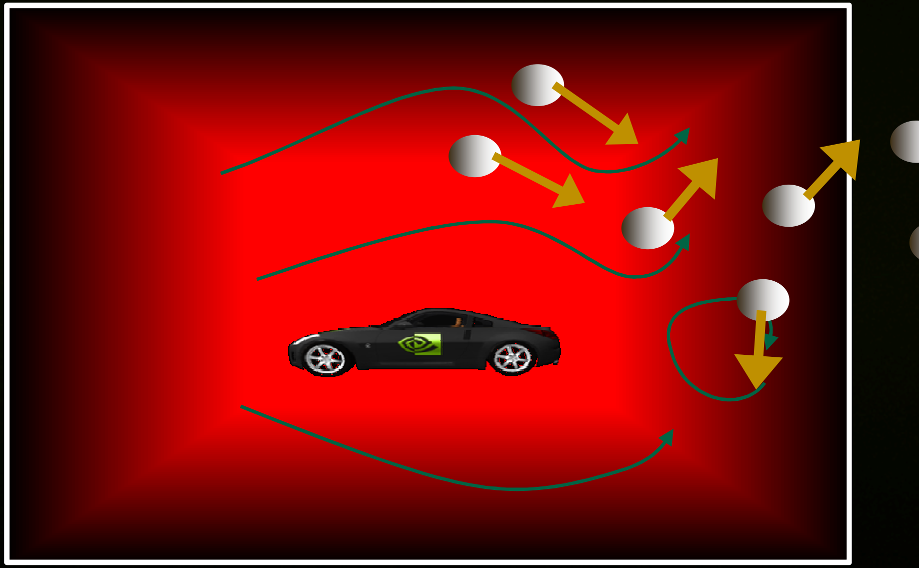
Calculate near-field fluid on grid
Fluid velocities drive particle ● motions

1. Calculate Fluid Velocities $\rightarrow$ on Regular Grid
   2nd-Order Accurate CUDA Multigrid Solver

2. Interpolate Fluid Velocities $\nearrow$ onto Particles
   3D Interpolation in CUDA

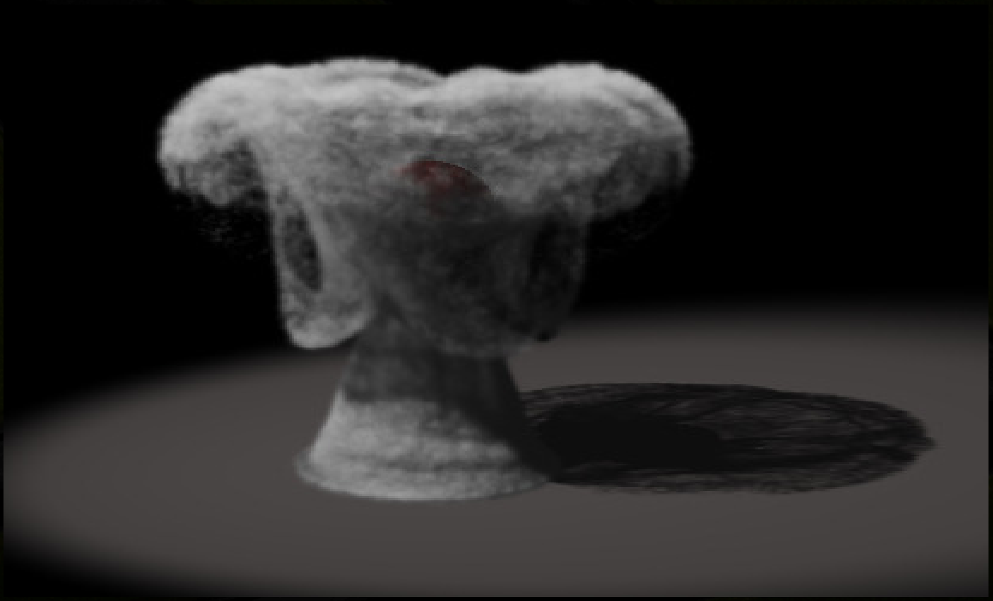
3. Advance Particles
   CUDA Particle System

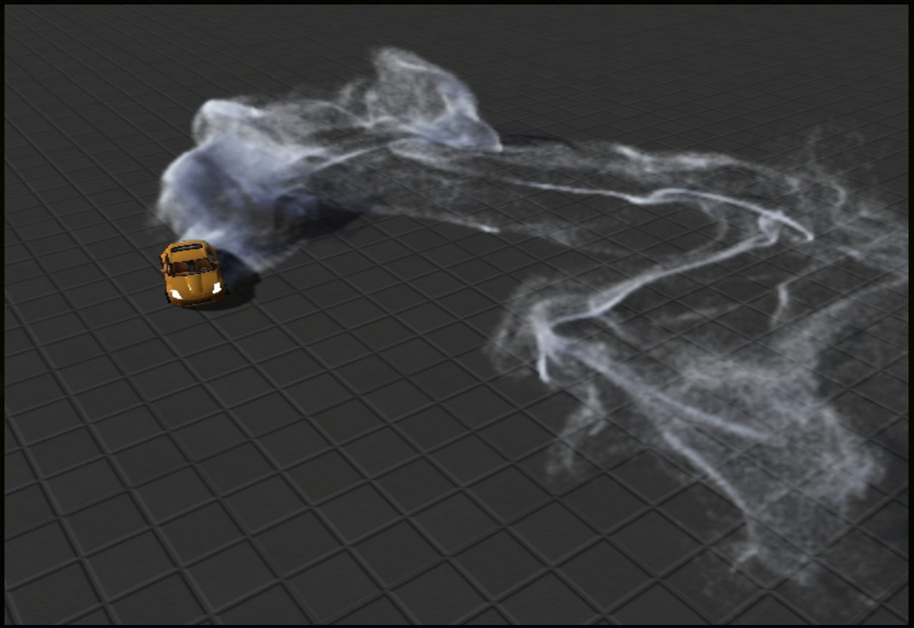
4. Render Particles
   CUDA - OpenGL Interop



APEX Turbulence

Interactive CFD Solution + Volume Rendering





Interactive Fluid-Particle Simulation
Using Translating Eulerian Grids.

Co-Processing

The Right Processors for the Right Tasks

2015 Projection

CPU-Alone	1.2^6	3X
CPU+GPU	50 * 1.5^6	570X

Heterogeneous Parallel Processing will (continue to) Fundamentally Change the way we do Graphics